第5章

ARMプロセッサでオーディオ・オシロスコープを実現する

――本誌2006年3月号付属ARM7基板による音声処理応用事例

山崎尊永

ここでは，本誌2006年3月号に付属したARM7基板を利用してオーディオ・オシロスコープを試作した例を紹介する．音声データをSDカードに記録し，合わせて波形をグラフィックスLCD（液晶ディスプレイ）に表示する．なお，部品表やマイコンの端子割り当て表，プログラム，本稿の関連コラム注1が，本誌付属のCD-ROMに収録されている．（編集部）

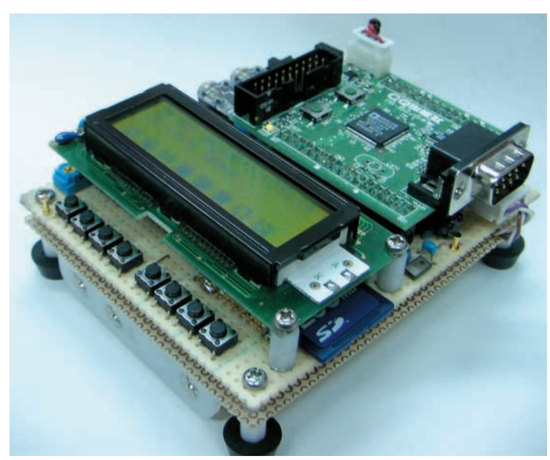
ARM7TDMIをコアにした米国Analog Devices社のアナログ・マイクロコントローラ（同社いわく，MicroConverter）の「ADuC7026」を搭載した基板が本誌2006年3月号に付属しました．本稿では，このADuC7026基板を用いた応用例を紹介してみたいと思います．

ADuC7026は，12ビットのA-DコンバータとD-Aコンバータをはじめ，各種周辺機能が集積されており，また，処理性能もそれなりに高い（動作周波数は41MHz）ので，これらの機能と性能を活用できる応用としてボイス・レコーダを製作してみます．単なるボイス・レコーダではおもしろくないので，音声データはSDカードにFAT（file allocation table）16形式で記録し，また，波形をグラフィックスLCDに表示できるようにしました．題して，「オーディオ・オシロスコープ」です．**写真1**に外観を示します．

注1：以下の四つのコラムが付属CD-ROMに収録されている．「Cのプログラム構造とリンカ・スクリプト」，「可変長記憶割り当て（malloc()など）の利用」，「割り込みフラグの注意点」，「コンパイルした結果を逆アセンブルしたい」．



（a）上から見たところ



（b）なゝめ上から見たところ

写真1　オーディオ・オシロスコープの外観

電子工作の塊．しかし．ソフトウェアの開発が全工数の95%を占めた．電源投入時に「CQ」のロゴを出す．LCDモジュールの下にSDカード・ソケットを設置した．

KeyWord　ARM7，LCD，SDカード，FAT16，ADPCM，ADuC7026，SPI，SG12232C，MEMS，I²C，NJM2073D，RTC-8564NB

マイコンの応用においては，ハードウェア周りの設計に対して，ソフトウェアの開発がかなりの工数を占めます．本稿では，ソフトウェアに関する部分も詳しく説明したいと思います．関連資料やソース・プログラム一式はすべて，本誌付属のCD-ROMに収録されています．

● ステレオ12ビット音声データをFAT16形式で記録

製作する本機の仕様を**表1**に示します．

サウンドは，マイコンが内蔵するA-D/D-Aコンバータの仕様に合わせてステレオ左右それぞれ12ビットです．サンプリング周波数はボイス・メモ程度の音質でよいので8kHzとしました．このステレオ・サウンド（L＋Rで24ビット）は，ADPCM（adaptive differential pulse code modulation）で8ビットに圧縮して記録します．

記録媒体はSDカードです．ADuC7026にはSDIOインターフェースはありませんが，SDカードにはSPI（serial peripheral interface）モードがあるので，ADuC7026が内蔵しているSPIで簡単にアクセス可能です．記録フォーマットは，SDカードをフラッシュ・メモリとして考えて独自フォーマットで読み書きしてもよかったのですが，今後のプログラムの再利用性も考えてFAT16フォーマットにしました．なお，誤操作による消去防止のため，本機ではデータ消去はサポートしなかったので，ファイル削除やメディアのフォーマットはパソコン上にSDカードをマウントして行ってください．

表示器は，台湾Sunlike Display Technology社の122ドット×32ドットのグラフィックスLCDモジュール「SG12232C」を使います．これをADuC7026の外部バスに接続して，サウンド波形をはじめ，各種表示を行います．

カレンダ時計チップをマイコンの外に置きました．バッテリ・バックアップによりメイン電源がOFFでも時計は動作し続けます．SDカードにFAT16ファイルを作成するとき，ディレクトリにファイル作成日時を記録しますが，その情報はこのカレンダ時計チップから入手しています．

電源はAC（交流）アダプタとバッテリの2電源をサポートします．全体の消費電流が少ないので，単3アルカリ乾電池で長時間動作が可能です．

表1　オーディオ・オシロスコープの仕様

12ビット・ステレオ・サウンドを1バイト/サンプルに圧縮している．サンプリングは8kHzなので，1秒当たり8000バイトの容量になる．256MバイトのSDカードで，約8～9時間程度録音できるだろう．

番　号	項　目	内　容
1	サウンド入力	ステレオ・マイク内蔵 外部LINE入力 12ビット，8kHzサンプリング
2	サウンド出力	ステレオ・スピーカ内蔵 外部ヘッドホン出力 12ビット，8kHzサンプリング
3	CODEC	ADPCM圧縮・伸張 （12ビット・ステレオ・サウンド←→ 8ビット・データ）
4	記録媒体	SDカード（SPIモード） FAT16フォーマット
5	表示器	122ドット×32ドットのグラフィックスLCD， カレンダ・時計表示，状態表示， 波形表示，キー・メニュー表示
6	カレンダ時計	バッテリ・バックアップ
7	キー入力	8個
8	電源	外部ACアダプタ（5V） または単3アルカリ乾電池×4本
9	使用マイコン	Analog Devices社製 ADuC7026

1．ハードウェアの製作

本稿の前半では，オーディオ・オシロスコープのハードウェアを中心に解説していきます．

● システム構成：端子割り当てに緊張

図1に本機のシステム構成を示します．アナログ入出力回路はアンプなどの部品が必要ですが，ADuC7026の豊富な周辺機能と5Vトレラント端子のおかげで，そのほかのグラフィックスLCDモジュールや，RTC（real time clock），SDカードはほとんど外付け部品なしで直接，接続可能です．

ところで，ADuC7026のような汎用マイコンを自分のシステムに応用する場合，端子割り当てがけっこう緊張する場面になります．独自にASIC（application specific integrated circuit）を作るのであれば自由に必要な端子機能を定義できますが，汎用マイコンの場合は端子機能があらかじめメーカによって決められてしまっています．とくに，最近の汎用マイコンは内蔵機能が豊富な反面，コストを抑えるためになるべく小ピンのパッケージに収めようとするので，1本の端子にたくさんの機能がマルチプレクスされていて，結果的に使えない内蔵モジュールがけっこうあります．

ADuC7026も，1本の端子に最大4機能が割り振られていてすべての内蔵モジュールの機能を使い切ることはできません．こうした情報は，マイコン・メーカのプレゼンテーションや概要資料には現れないケースが多く，マイコンを選定する際には，じっくりデータシートを見る必要があります（じつは，当の半導体メーカのマイコン開発者も，マ



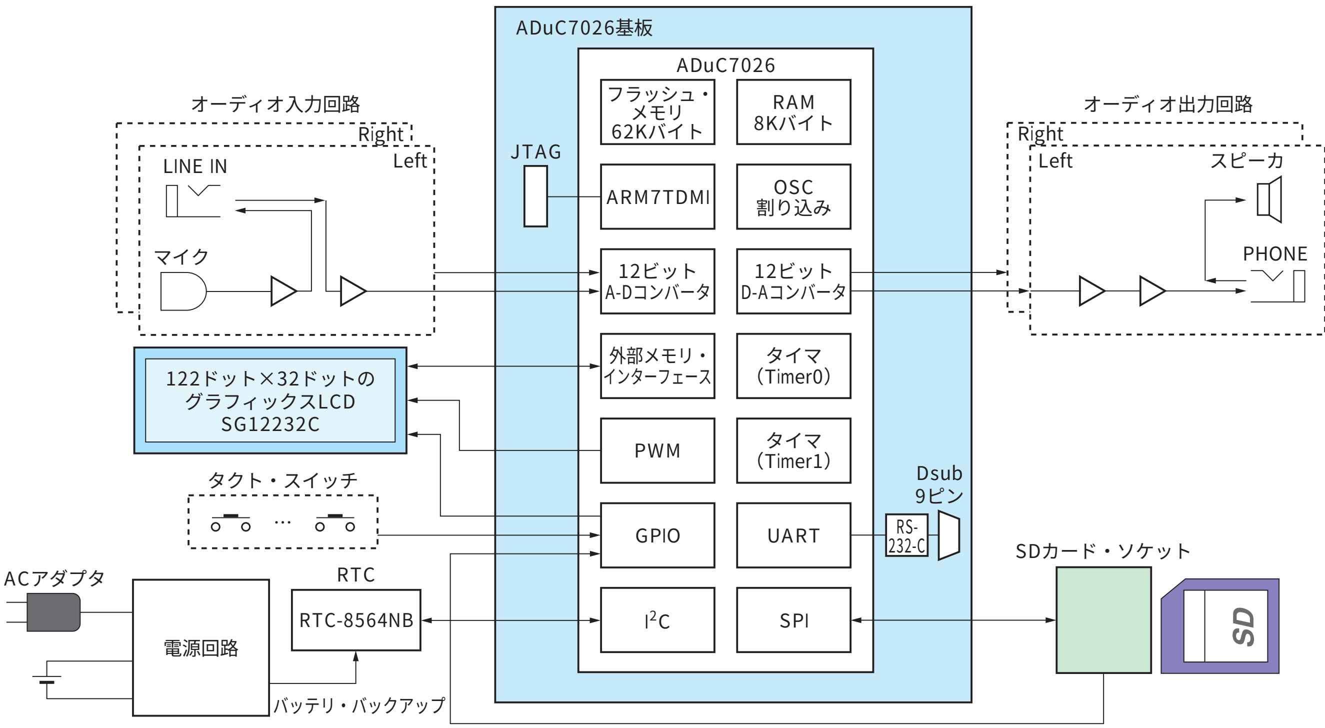


図1　オーディオ・オシロスコープのシステム・ブロック図
GPIOは，1）グラフィックスLCDのリセット，2）グラフィックスLCDのバックライト制御，3）タクト・スイッチ入力，4）SDカード・ソケットの挿入/LOCK検知入力に使用している．グラフィックスLCDのシステム・クロック（2kHz）はPWMで発生させている．ADuC7026の豊富な周辺機能のおかげで，このようにシンプルな構成にできた．

イコン製品の端子機能割り当てにはおおいに苦労している．なるべく多くの応用分野に適合できるように努力しているが，個々のユーザによってはかなり不満な端子機能になっていることがあるからだ）．

本機に使用するADuC7026の端子機能割り当ては，本誌付属のCD-ROMに収録されています（ADuC7026端子割り当て.pdf）．端子機能の設定方法と回路上の接続先を記載しました．ADuC7026はすべてのGPIO（general purpose input/output port）端子が5Vトレラントであり，5V電源を使うRTCチップとLCDモジュールの接続もスムーズです．

なお，内蔵プルアップMOSはすべてのGPIOに付いていますが，ON/OFFを制御できるのはP0の一部とP1のみです．もし，ADuC7026をSTOPモードにして，消費電流をリーク電流のみに抑えようと思う場合，プルアップMOS経由で外部回路に流れ出す電流に注意する必要があります．

● 回路構成：入手しやすいMEMSやLCDを利用

本機の回路図を図2に示します．部品表は，本誌付属のCD-ROMに収録されています（部品表.pdf）．

1）電源回路

電源はまず，製作するメイン・ボードにACアダプタまたはバッテリから5V電源DVDD5を供給し，さらにここからADuC7026が載っている基板へ電源を供給します．コネクタCN4は，マイコン・ボードへの電源供給と，バッテリ接続を兼ねています．ダイオードD2とコンデンサC7は，バッテリ交換時などでもRTCチップに電源供給を継続するためのものです．RTCチップの消費電流は，I^2C非アクセス時は330nA程度（標準値）なので，電池がなくても容量だけでかなりの時間，動作します．なお，この回路だとD2に流れる電流が非常に小さいので，D2の電圧降下は0.1V程度しかなく，電池が新品（6V以上）の場合，RTCチップの電源電圧の推奨動作条件である最大5.5Vを超えることがありますが，とくに問題なく動作しています．最大絶対定格の6.5Vは超えないので，壊れることはありません．

また，DC（直流）ジャックCN1のスイッチ機構により，ACアダプタが接続されているときにバッテリからの接続をOFFにしています．ACアダプタをコンセントに差さずにおくと，やがてRTC電源が落ちて時計がクリアされてしまいます．AC側からの電源とバッテリ側からの電源を

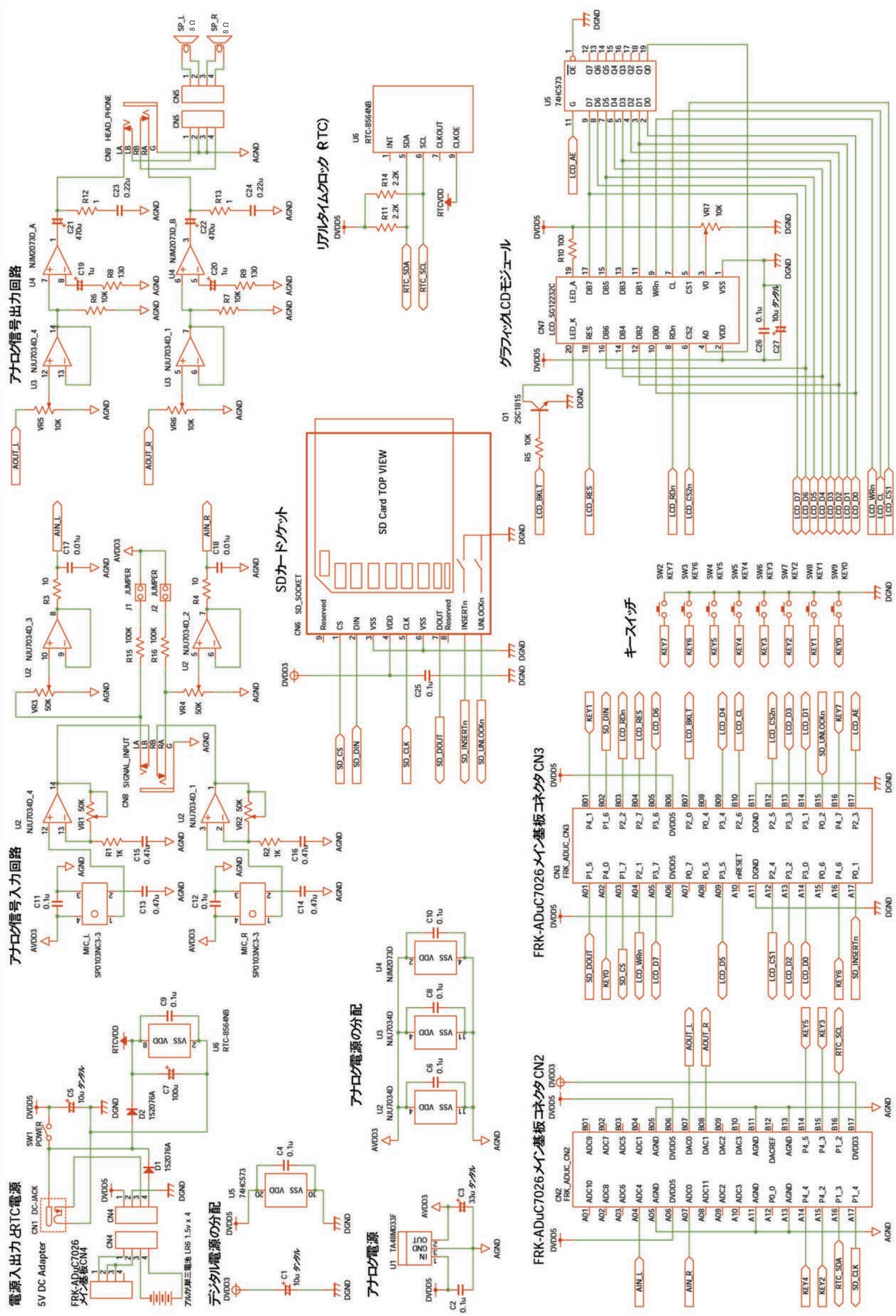


図2 本機の回路図
使用している部品については，本誌付属のCD-ROMに収録されている部品表（部品表.pdf）を参照．

ダイオードORしてもよいでしょう．

2）アナログ信号入力回路

アナログ信号入力回路では，マイクからの音声を増幅し，LINE入力のジャックを通してA-Dコンバータに入力します．ステレオ（L，R）2回路を用意します．A-Dコンバータはシングルエンド・モードで使用し，リファレンス電源は内蔵のバンド・ギャップ電源（2.5V）を用います．よって，A-Dコンバータへの入力電圧と変換コードの対応は，0V入力時に0x000で，2.5V入力時に0xFFFとなります．オーディオ入力回路で，A-Dコンバータへの入力波形が0V～2.5Vの範囲に収まるように調整します．

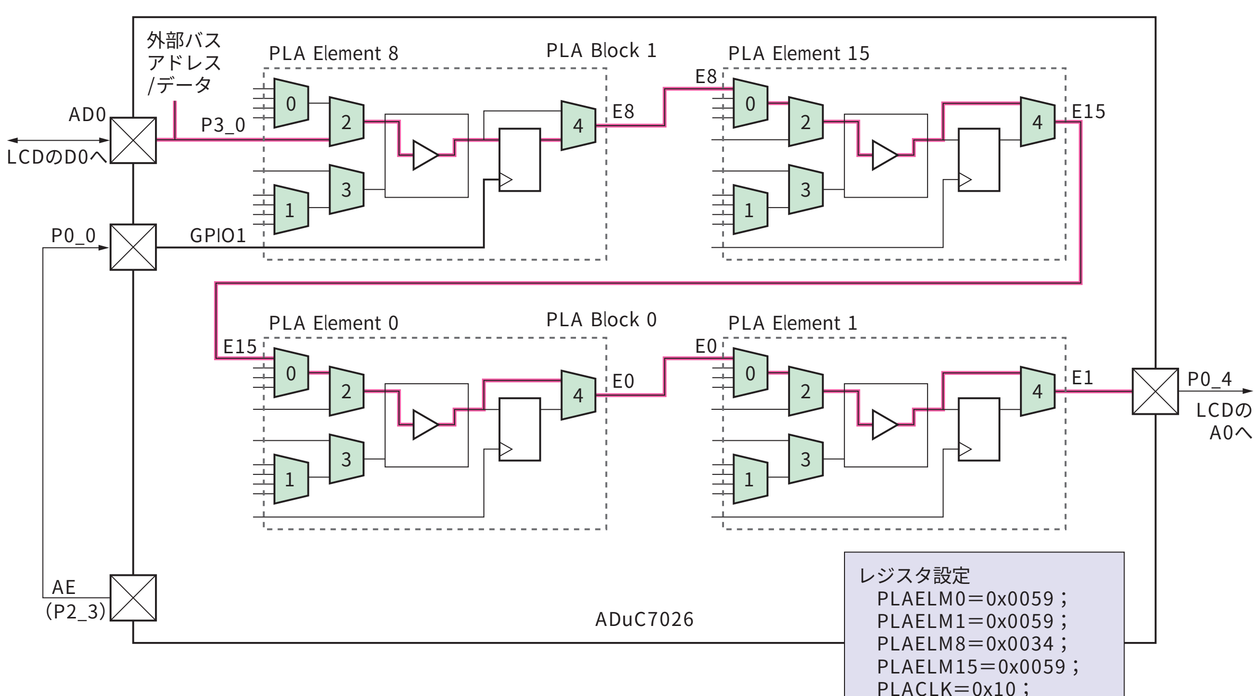
マイクは，米国Knowles Acoustics社の小型MEMS（micro electro mechanical systems）シリコン・マイク「SP0103NC3-3」を使用しました．筆者は秋葉原の秋月電子通商で入手しました．中にアンプが内蔵されていて，外部にコンデンサC13とC14を付けることで20dBのゲイン（利

Column 内蔵PLAをどう使おうか？

ADuC7026はPLA（programmable logic array）を内蔵しています．16個のPLAエレメントから構成される小規模なもので，1個のエレメントには2入力ルックアップ・テーブルとフリップフロップが1個ずつ入っています．HDLから直接合成することはできず，自分でルックアップ・テーブルの設定やフリップフロップの接続（マルチプレクサ）を選択します．このためのツール「PLA Tool」が開発元から提供されており，GUI（graphical user interface）を使って回路接続を選択していけば，レジスタ設定を出力してくれます．かなり小規模なPLAですが，ちょっとした基板上のグルー論理を実現するのに使えそうです．PLAの入力については，外部端子からPLA専用に入力する方法と，端子のレベルそのもの（GPIO）を入力する方法から選択できます．端子機能をPLA入力以外に設定していても，その端子機能の信号をPLAに入力でき，かなり便利に使えます．

本機でもこのPLAを活用すると，外部マルチプレクス・バスのアドレス分離に用いたラッチ（74HC573）が不要になります．LCDにとって必要なアドレスは1本だけなので，このPLAでラッチを代用できます．**図A-1**のように74HC573を削除して一部配線を追加するだけです．レジスタ設定は，init.cのGPIO設定後に入れてください．

本機では，PLA Elementを4個使用することで，外部の74HC573が不要になります．PLA Element 8でAD0をAE（の立ち上がりエッジ）でラッチします．残りのPLA Elementは，ラッチしたアドレスA0をP0_4端子から出力するための配線経路として使っています．なお，**図2**の外部回路を次のように修正してください．1）74HC573を外す，2）AE（P2_3）をP0_0に接続する，3）P0_4をLCDのアドレスA0に接続する．なお，PLAを使って，P4_0～P4_7から分離（ラッチ）したアドレスを出力するようにすると，P3_xをデータ（8ビット），P4_xをアドレス（8ビット）とする本格的なアドレス・データ分離バスを実現できます．本機ではP4_xをスイッチ入力に使っているので，この方式はとれませんが，さまざまなアプリケーションで利用できると思います．



図A-1　内蔵PLAの活用

得）が得られます．この出力を単一電源OPアンプで増幅します．

単一電源OPアンプをDC的に使おうとすると，波形が0V以下でクリップされないように，少し複雑な計算を行って動作点を決め，外付け抵抗の回路構造とパラメータを決定する必要がありますが，ここではコンデンサC15とC16で動作点を持ち上げる方法をとりました．半固定抵抗VR1とVR2でゲインを調整します．

CN8はLINE入力のジャックです．プラグを挿すとマイクからの信号が切れるようにします．なお，AC結合のLINE信号を入力する場合，そのまま後段の単一電源OPアンプに入力するとGNDよりもマイナス側の波形がクリップされてしまうので，ジャンパJ1とJ2を短絡することでDC的に浮かせられるようにします．半固定抵抗VR3とVR4は，このLINE入力とマイク入力の振幅を調整するものです．

3）アナログ信号出力回路

アナログ信号出力回路では，D-Aコンバータからの出力をパワー・アンプで増幅して，ヘッドフォン・ジャック経由でスピーカに与えます．これもステレオ（L，R）2回路を用意します．D-Aコンバータのリファレンス電源は，バンド・ギャップ電源（2.5V）とアナログ電源（3.3V）のどちらでもよいのですが，本機では2.5Vを用いました（ソフトウェアによる設定）．入力コードとD-Aコンバータの出力電圧の対応は，0x000入力で0V，0xFFF入力で2.5Vとなります．最大振幅が3.3Vになることを想定して回路を設計します．

回路は簡単で，D-Aコンバータ出力を半固定抵抗VR5とVR6により振幅調整したあと，最終的に新日本無線のパワー・アンプ「NJM2073D」で増幅して，ヘッドホンおよびスピーカに供給します．途中にあるボルテージ・フォロアのOPアンプは試作途上で入れたものなので，なくてもかまいません．NJM2073Dは発振しやすいので，状況に応じてパラメータを調整する必要があります．NJM2073Dのデータシートの推奨回路例にはR8とR9はないのですが，付けたほうが安定しました．あとは，（－）入力端子とAGND間にコンデンサを入れて発振を防止する方法があります．

なお，R12/R13，C23/C24のパラメータは使用するスピーカに応じて調整する必要があります．ないほうが安定する場合もあるようです．

4）グラフィックスLCDモジュール

LCDモジュール「SG12232C」は秋月電子通商で入手しました．ADuC7026とは8ビット幅でバス接続しますが，ADuC7026がマルチプレクス・バスなので，アドレスをDラッチ（74HC573）で分離します．LCDが使用するアドレスは1本だけですが，今後の回路拡張を考えて8ビットのDラッチを使いました．

このLCDのバス・タイミング仕様は，80系方式，68系方式のいずれかを選択できるようになっています．ADuC7026の外部バスは基本的に80系のタイプなので，LCDモジュールのリセット端子（/RES）を“L”→“H”→“L”と変化させ，そのまま“L”に保持して80系のタイミングに設定します．この/RES信号はADuC7026のGPIOから与えます．

このグラフィックスLCDのサイズは122ドット×32ドットですが，左の61ドット×32ドットの面と右の61ドット×32ドットの面のそれぞれを，個別の制御チップ（セイコーエプソンの「S1D15200」2個）がコントロールしています．このため，このLCDにはチップ・セレクト（CS）が2本あります．LCDモジュールの各レジスタの外部メモリ空間上のアドレス配置を**表2**にまとめておきます．

また，このグラフィックスLCDには，2kHz程度のクロックを常時与える必要があります．適当な発振器を用意してもよいのですが，ADuC7026のPWM（pulse width modulation）から与えることにしました．このPWMモジュールは，ノンオーバラップ期間をもつ3相PWM波形を出力できる本格派で，ACモータのインバータ制御が可能なものです．少しもったいない使いかたですが，2kHzの50％デューティ・クロックを生成させるためだけに使います．

なお，このグラフィックスLCDモジュールはフォントを内蔵していないので，文字を表示させる場合は，ユーザ自身で文字データを用意する必要があります．このため，本機には，アルファベット・数字・記号のASCII文字と一部ユーザ外字を作成してソフトウェアに含めてあります．

表2 グラフィックスLCDモジュールのアドレス配置

番　号	LCD面	割り当てエリア	レジスタ		アドレス
1	左側 61ドット×32ドット	MS0	コントロール	CTRL	0x1000_0000
			データ	DATA	0x1000_0001
2	右側 61ドット×32ドット	MS1	コントロール	CTRL	0x2000_0000
			データ	DATA	0x2000_0001

半固定抵抗VR7はコントラスト調整です．またLCDのバックライトLEDのON/OFFを，トランジスタQ1を介してGPIOから制御できるようになっています．

5）SDカード

SDカードは，本来のSDIOインターフェースでアクセスしたほうが4ビット・パラレルで送受信できるので高速です．しかし，ADuC7026にはそのようなインターフェースがないので，SDカードのSPIモード（シリアル）を用いてアクセスすることにします．SDカードのSPIモードはMMCカードのSPIモードと互換性があるので，本機ではMMCカードも使えると思います．MMCカードは物理的にもSDカードのソケットに挿すことができます．また，SPIモードのピン配置も同じです．なお，試作機では米国SanDisk社製の256Mバイト・カード（SDSDB-256-J60）を用いてデバッグしており，ほかのカードの互換性チェックは十分に行っていないので，本機を製作される場合，まずはSanDisk社製カードの使用をお勧めします．

SDカード・ソケットには，カードが挿入されたことを検知するスイッチと，LOCKスライドの検知スイッチがあります．これらはそのままGPIOに直結しておきます．試作機ではSDカード・ソケットとして，単体のジャンク品を使用しましたが，サンハヤトからもSDメモリ・カード用変換基板（CK-29）が出ています（**写真2**）．いずれの製品も，挿入検知スイッチとLOCKスライド検知スイッチが付いているので，どのピンがそれに相当するか，SDカードの実物を抜き差しして調べてみてください．

6）カレンダ時計

カレンダ時計用のRTCチップには，セイコーエプソンの「RTC-8564NB」を使います．これはI^{2}Cインターフェースでホストと通信できるタイプです．ADuC7026のI^{2}Cモジュールから制御します．メイン電源が切れていても，バッテリから電源供給が続くようにします．

RTC-8564NBは表面実装パッケージ品ですが，これを8ピンDIPに変換する基板を秋月電子通商で入手できます．**図2**の回路図上ではRTC-8564NBそのもののピン番号を記載してあるので，秋月電子通商の交換基板を使用する場合は，説明書に従って配線してください．変換基板上にプルアップなどを行えるジャンパがありますが，いずれもオープンにしておいてください．I^{2}CインターフェースのSDA信号，SCL信号は外部回路でDVDD5にプルアップします．

7）キー・スイッチ

ユーザ操作用のタクト・スイッチを8個，用意しています．チャタリング除去などの処理はソフトウェアで行うことにして，直接GPIOに接続しています．プルアップ抵抗はADuC7026内蔵のものを使っています．

● 基板の製作：とにかく楽しみながら作ろう

いよいよ基板の製作です．付属CD-ROMに収録された部品表（部品表.pdf）を参考に，部品を集めます．なお，こうした基板作りで気を使うのが，コネクタやスイッチなどの機構部品です．実現したい外形や構造をイメージしながら，楽しく部品を集めましょう．

1）ADuC7026基板の改造

改造というほどでもありませんが，ADuC7026基板はデフォルトのままではI^{2}Cインターフェースが使えません．次の加工を施してください（**写真3**）．

● JP1をパターン・カット

一見，切れているように見えますが，じつは細いパターンでつながっています．これをカッターでパターン・カットします．

● JP2をはんだで短絡させる

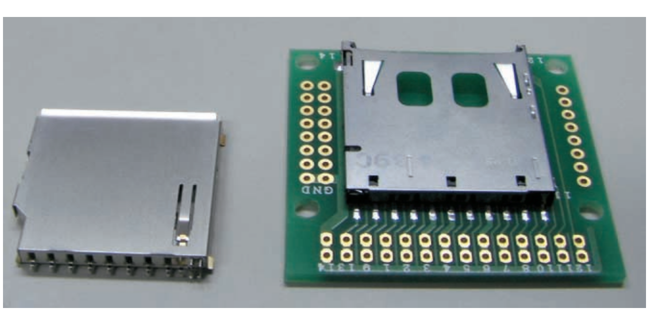


写真2　SDカード・ソケット
左は，単体ソケット．実装面積が小さい．試作機ではこちらを使った．右は，サンハヤトのメモリ・カード用変換基板「CK-29」．プッシュ・プッシュ排出タイプなので使いがってがよい．

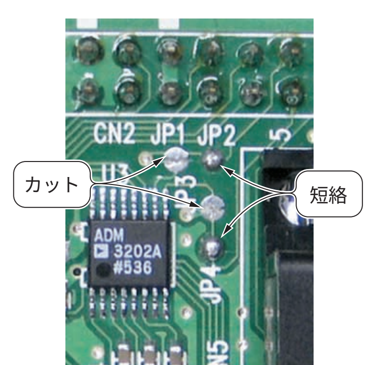
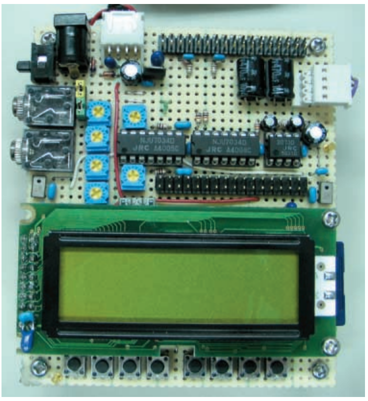
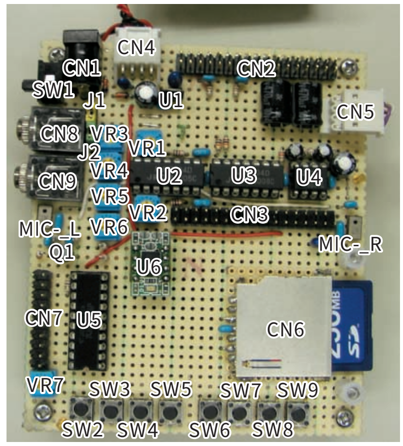


写真3
ADuC7026基板の改造
I^{2}Cを使うために，JP1とJP3をカットし，JP2とJP4を短絡させる．



（a）ADuC7026基板を外した状態



（b）さらにLCDモジュールを外した状態

写真4　試作機のメイン・ボード

SDカード・ソケットは，2.54mmピッチのランドが並んだシール基板をボンドではり付けて，その上に実装した．

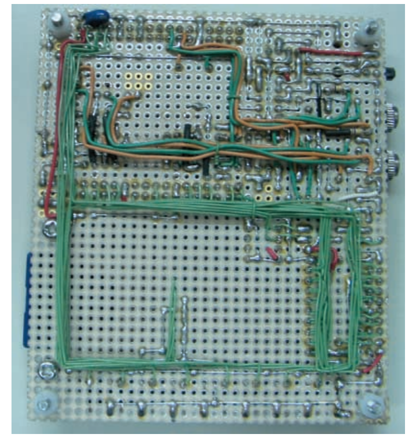


写真5　試作機のメイン・ボードの配線

はずかしい試作基板の配線面．まず，電源をスズめっき線を主体にアナログとディジタルを分離して配線する．次に，アナログ信号をリード線で配線し，最後にディジタル信号をラッピング線で配線した．

● JP3をパターン・カット

JP1と同じです．

● JP4をはんだで短絡させる

この改造によって，UARTのRTS/CTSとRS-232-Cコネクタの間が切れますが，ハードウェア・フロー制御を行わなければとくに問題はありません．UART経由のフラッシュ・メモリの書き込みも問題なく行えます．

2）メイン・ボードの製作

ユニバーサル基板上に部品を配置して配線していきます．配置例を**写真4**に示します．

SDカード・ソケットは，2.54mmピッチのランドが並んだサンハヤトのシール基板「ICB-059」をソケットのサイズよりひと回りくらい大きめに切って，メイン・ボードにボンドではり付け，その上にはんだで実装しました．SDカード・ソケットには一部ピッチが狭いピンがありますが，そこにはラッピング線を直接はんだ付けします．

メイン・ボードの配線（**写真5**）は，次の順番で行います．

● 電源の配線

まず，ディジタル・グラウンドDGND，アナログ・グラウンドAGNDを分離して，きっちりスズめっき線などで配線します．次に，電源系（DVDD5，RTCVDD，DVDD3，AVDD3）を配線して，必要なバイパス・コンデンサを接続します．配線が終わったら電源間の短絡がないかなどをよく確認し，いったん電源を入れてみて，各ポイントに正しい電圧が印加されているかどうかをチェックしてください．3端子レギュレータU1（TA48M033F）は中央のリードが3番ピン（GND）なので注意してください．

● アナログ信号の配線

アナログ入力回路，アナログ出力回路を実装・配線していきます．アナログ回路が完成したら，簡単に動作をチェックしましょう．AIN_LとAOUT_L，AIN_RとAOUT_Rをそれぞれ直結してみて，ヘッドホンをCN9に差して，マイクに入力された音がヘッドホンから聞こえるかどうか，VR1～VR6を調整しながら確認してください（ヘッドホンなしでスピーカを鳴らすとハウリングを起こしてしまうので要注意）．前述したように，とくにU4のパワー・アンプは発振しやすいので，パラメータの調整が必要になることがあります．

● ディジタル信号の配線

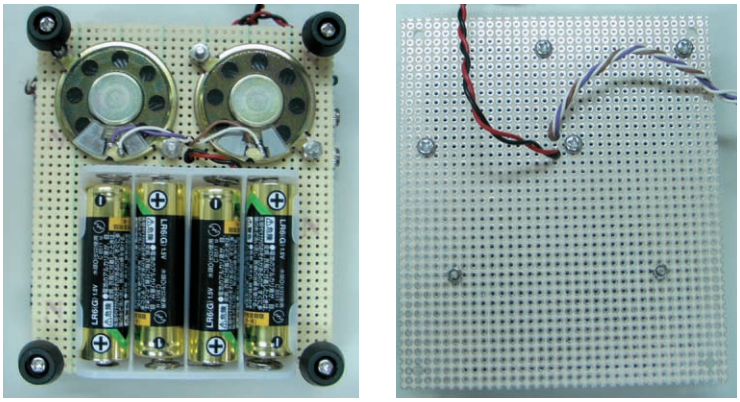
電源とアナログ信号が終われば，95％は完成したようなものです．あとはミスのないようにじっくりとディジタル信号を接続しましょう．

3）スピーカ・バッテリ基板の製作

これは**写真6**に示したようなスピーカと電池ボックスだけを搭載した基板で，試作機ではメイン・ボードと背中合わせに取り付けたものです．このあたりは，各自で実現したい構造に応じて作成してください．

● 調整と動作テスト：あせらず順番にチェック

基板が完成したと思ったら，ここでいったん落ち着いてコーヒーを1杯飲みましょう（定番のコメント）．そしてあらためて，全体の配線をチェックします．



（a）下から見たところ　（b）上から見たところ

写真6　試作機のスピーカ・バッテリ基板
スピーカと電池を搭載した基板を，メイン・ボードの裏側に背中合わせに取り付けた．

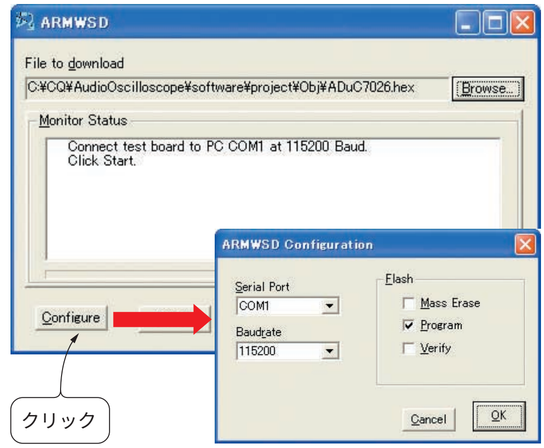


図3　内蔵フラッシュ・メモリへのダウンロード・ツールARMWSD
ARMWSDツールは，ADuC7026の内蔵フラッシュ・メモリへバイナリ・コードをダウンロードするツールである．[Configure]ボタンを選択し，ボーレートを「115200bps」に設定する．

1）簡単な動作のチェック

ADuC7026基板はまだ載せずに，電源を入れてみて再度各部の電圧をチェックします．異常に熱くなったりする部品がないこともチェックしましょう．そして，トイレに行って落ち着いてから，大事なADuC7026基板を載せます．ここで電源を入れて，ADuC7026基板の発光ダイオードD2が点灯することを確認してください．念のため，SW2をONにするとD1が点灯することも確認してください．

2）アナログ入力回路の調整

手持ちのパソコンから何か音を出してください．ニュースのアナウンスでも，音楽でもけっこうです．便利なのは，フリーで出回っている正弦波の音を出してくれるソフトウェアです．これらを鳴らしながら，マイクで音を拾わせて，A-Dコンバータの入力端子（L，R両方）をオシロスコープで見ながら，0V～2.5V付近の間でいっぱいいっぱいに波形が振れるように，VR1～VR4を調整してください．ゲインを大きくし過ぎると波形が飽和（クリップ）してしまうので注意してください．オシロスコープが手元になければ，マイコンにソフトウェアをダウンロードしてから実際に録音・再生し，聞こえ方で判断して調整してもかまいません．

3）アナログ出力回路の調整

最初はVR5とVR6は絞っておいてください．マイコンから音が出るようになったら，適度に聞こえるように調整してください．

4）ソフトウェアのダウンロード

さて，マイコンは「ソフトなければただの石」です．缶コーヒーみたいなものです．ハードウェアは缶に相当して，それがないとモノを実現できない重要なものですが，より大事なのは中身のコーヒー（ソフトウェア）です．ソフトウェアがシステムに命を吹き込んでくれます．

動作チェックのため，コンパイル済みのバイナリ（コード）をダウンロードしてみましょう．ダウンロードするのは，付属CD-ROMに収録されている以下のものです．

AudioOscilloscope\software\project\Obj\ADuC7026.hex

これを，Analog Devices社提供のツール「ARMWSD」を使い，ADuC7026の内蔵フラッシュ・メモリに書き込みます（図3）．パソコンとマイコン基板の間は，RS-232-Cストレート・ケーブルで接続します．プログラム・サイズが約40Kバイトあるので，転送時間短縮のため，［Configure］ボタンを押してARMWSD Configurationダイアログを開き，「115200bps」に設定してください．マイコン側も自動的に最初の通信ネゴシエーションで115200bpsに合わせてくれます．次に［Browse...］ボタンで上記ファイルを選択し，［Start］ボタンを押してください．その状態で，ADuC7026基板のSW2（ブート・モード）を押しながらSW1（リセット）を押して離します．最後にSW2を離します．これでフラッシュ・メモリへの書き込みが行われます（何度も書き込みを行っていると，まれにうまく書き込まれず，暴走することがある．ARMWSD Configurationダイアログで「Verify」にチェックを入れると安心）．

さて，いよいよ起動です．「Run」を押してください．LCDに「CQ」のロゴや何らかのメッセージは出力されましたか？ 見えなければVR7でコントラストを調整してみて

ください．うまくいけばそれ以降は，電源を投入するか，ADuC7026基板のSW1を押せば起動します．

● 使用方法：LCD表示を確認しながらキーを操作

先ほどダウンロードしたバイナリによる本機の動作を**写真7**で説明します．最初は，RTCチップが低電圧検出状態になっているので，**写真7(c)**に示す時刻設定画面になります．ここで時刻を設定してください．その後，**写真7(a)**の画面になり，まだSDカードを挿入していなければ，「No SD Card」と表示されます．ここでSDカードを挿してうまく認識されると，「SD Card Ready」と表示されます．なお，SDカードはあらかじめ，パソコン上でFAT(FAT16)形式でフォーマットしておいてください．FAT12やFAT32には対応していません．なお，大事なデータが入ったSDカードでは試さないようにしましょう．配線がまちがっていたら，電気的に壊れる可能性もあります．

メイン画面で録音キーを押すと，新規ファイル名を表示し，録音するかどうかを聞いてきます．そこで録音キーを押すと**写真7(d)**の画面になり，波形を表示しながらSDカードに録音していきます．SDカードへの記録は，

　\AUDIOOSC

というターゲット・ディレクトリの下に，

　00000001.AWV，00000002.AWV

というようなファイル名で保存していきます．ファイル名は8＋3形式で生成し，ロング・ネームのファイルは生成しません．録音開始時にSDカードのルート・ディレクトリ内にターゲット・ディレクトリ\AUDIOOSCがなければ，自動作成します．また，ルート・ディレクトリやターゲット・ディレクトリにほかのロング・ネーム・ファイルやディレクトリが存在していても問題ありません(無視する)．

再生キーを押すと，**写真7(e)**のようなファイル選択画面が出ます．ここで再生したいファイルを選択すると，**写真7(f)**のように波形を表示しながら再生が始まります．

録音したサウンド・データはパソコン上にバックアップしておけます．SDカード上のファイルの消去はパソコンの上で行ってください．

本機の消費電力は，回路全体でおおよそ以下のとおりでした．

- メイン・メニュー表示時(SD Card Ready)：70mA
- 録音時：70mA
- 再生時(スピーカ駆動)：100mA～130mA
- バックライトON時：10mA増加
- 電源OFF時(RTCのみ消費)：0.1μA以下

2000mAhの電池なら，15時間程度は連続再生できそうです．

2．ソフトウェアの開発

ここから先は本機のソフトウェアのしくみを解説し，本機の動作を改良したり変更したりするための情報を提供します．GNUによる組み込みソフトウェア開発のいくつかの側面についても解説します．

● ソフトウェア開発環境を立ち上げる

ADuC7026のソフトウェア開発環境の一つとして，Keil Software社(現在はARM社の傘下にある)の「μVision3」があります．評価版はダウンロードして無償で使えますが，評価用Keil純正コンパイラと組み合わせると，生成できる



(a) メイン画面　(d) 録音中
(b) ヘルプ画面　(e) 再生ファイルの選択画面
(c) 時刻設定　(f) 再生中

写真7　オーディオ・オシロスコープの使いかたとLCD表示例

リセット後，ロゴとメッセージを表示して，(a)の画面になる．最上段に時計を表示している．SDカードの挿入状況も示す．キーは左から，バックライト，ヘルプ，時刻設定，(三つ飛んで)録音，再生．

メイン画面からヘルプ・キーを押すと(b)の画面になる．8ページのメッセージを表示する．キーは左から，バックライト，QUIT，ページ戻り，ページ送り．

メイン画面から時刻設定キーを押すと(c)の画面になる．キーは左から，バックライト，(一つ飛んで)取り消し，設定，デクリメント，インクリメント．設定キーで，一個ずつ設定していく．秒のところで設定キーを押すと，RTCに時刻が設定される．

メイン画面で録音キーを押すと，ファイル名(先頭8文字)と録音するかどうかを聞いてくる．いちばん右の録音キーを押すと(d)の画面になる．録音中のLとRの波形を表示する．キーは左から，バックライト，(六つ飛んで)録音停止．

メイン画面で再生キーを押すと，(e)の画面になる．左からファイル名(先頭8文字)，ファイル作成日，ファイル容量を示す．キーは左から，バックライト，(三つ飛んで)上スクロール，下スクロール，選択(再生)，取り消し．

再生ファイルの選択画面でファイルを選択すると(f)の画面になる．音を再生しながら，録音時と同じように波形を表示する．キーは左から，バックライト，(五つ飛んで)再生停止．

オブジェクト・サイズに16Kバイトの制限が付いてしまいます．一方，GNU gccコンパイラと組み合わせることもできて，その場合は制限がありません．このため，本機はμVision3上でgccによる開発を行いました．

1）ツールのインストール

μVision3は，バージョンが上がるに従って，ADuC7026の各種定義ファイルなどが改善されてきているので，Keil社のサイト（http://www.keil.com/）の「Evaluation Software」から最新版をダウンロードしてください．原稿執筆時点（2006年2月）の最新ファイルは下記のとおりです．

- μVision3本体：rvmdk250a.exe
- ARM用gcc　：gccARM331.exe

これらをインストールしてください．この後，CD-ROMに入っている以下のプロジェクト・ファイルを開きます．

AudioOscilloscope\software\project\ADuC7026.Uv2

すると，図4に示すようにツールが立ち上がります．

2）ツールの設定

AudioOscilloscopeフォルダの中のプロジェクト・ファイルを立ち上げると，開発ツールはGNU用に設定された状態になります．自分でプロジェクト・ファイルを新規に作成するときは，あらためて設定し直す必要があります．GNUツールを使う場合の設定方法を図5，図6に示したので参考にしてください．

3）コンパイルの方法

プログラムのコンパイルは，メイン・ウィンドウ左側の「Project Workspace」の枠内の「Target1」を選択した状態で，「Project」→「Build target」を選ぶことで実行できます．リンクまで実行します．Build targetは変更した箇所に関係する部分だけ再コンパイルします．すべてコンパイルし直したいときは，「Project」→「Rebuild all target files」で実行できます．

さて，GNUといえばMakefileはどこにあるのか，ということが気になります．本ツールにはMakefileはなく，代わりに独自のバッチ・ファイルを生成して，コンパイルからリンクまでを行っています．

AudioOscilloscope\software\project\Target1.BAT

がそれで，各ソースのコンパイルとリンクを，コマンド・オプションが記載されたファイルを参照しながら実行するようになっています．

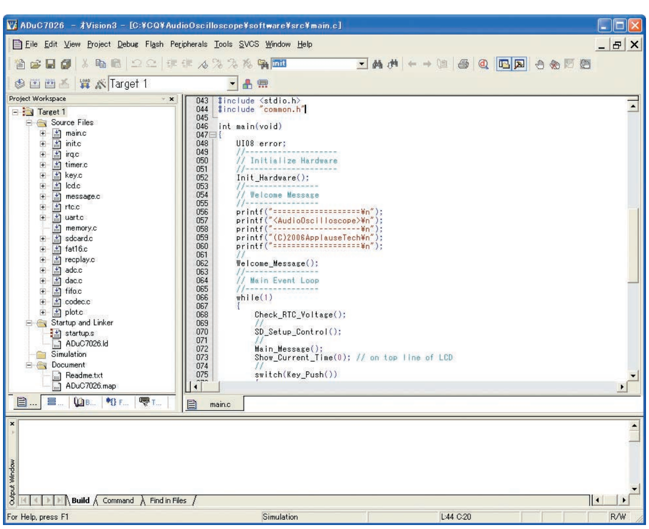
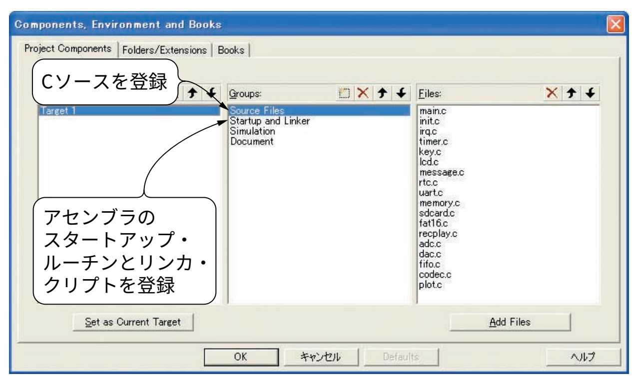
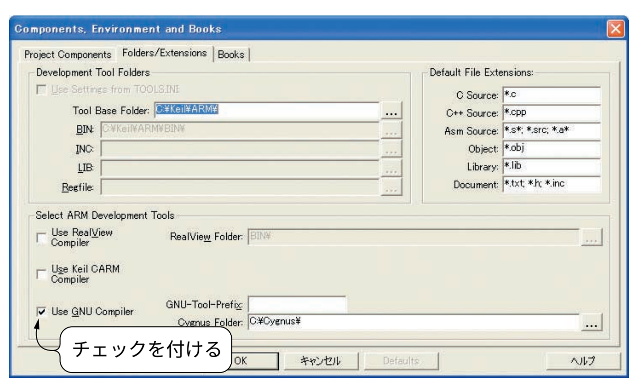


図4　Keil社のARM開発環境μVision3

本機のプロジェクト・ファイルAudioOscilloscope\software\project\ADuC7026.Uv2を立ち上げたところ．ツールのバージョンは，μVision3 V3.30a + gccARM311を使った．



（a）Project Componentsタブ



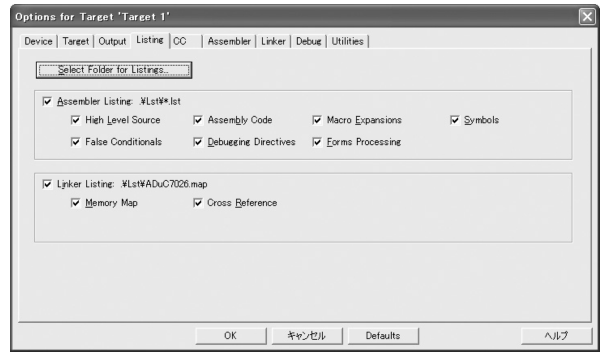
（b）Folders/Extensions

図5　Components, Environment and Booksダイアログ

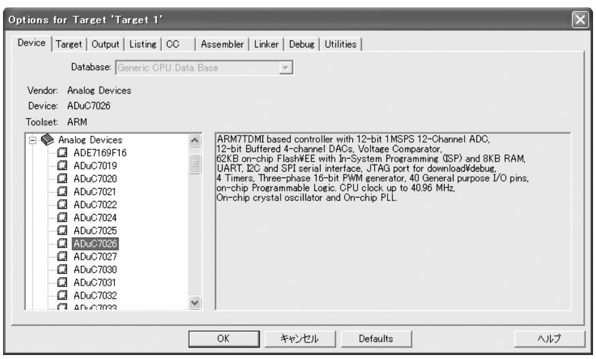
「Project」→「Components, Environment, Books」を選択して開く．Project Componentsタブ内では，GroupsのSource Filesに各Cソースを，またStartup and Linkerにアセンブラのスタートアップ・ルーチンとリンカ・スクリプトをそれぞれ登録する．Folders/Extensionsタブ内では，GNUツールを使用することを指定する．

● デバッグ環境を整備する

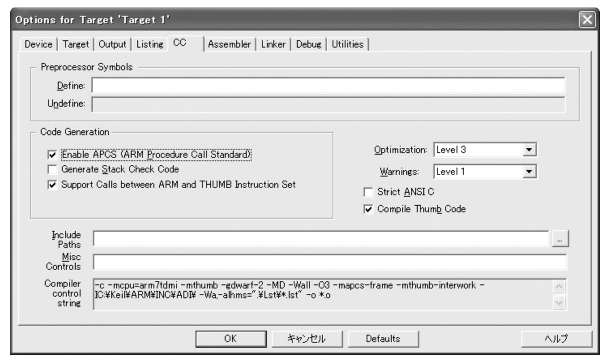
組み込みプログラムのデバッグというと，ICE（in-circuit emulator）を使うのが定番です．ADuC7026の基板にもJTAG ICEを接続するコネクタが付いていて，筆者も手持ちのARM対応ICEを接続して動作を確認できました．ただ，シンプルなJTAG ICEだと，ARM7TDMIが持っているオンチップ・デバッガを使うことになるので，例えばハードウェア・ブレーク（内部論理でイベントとブレーク条件を比較）の本数が少ない，というような制約が出てきま



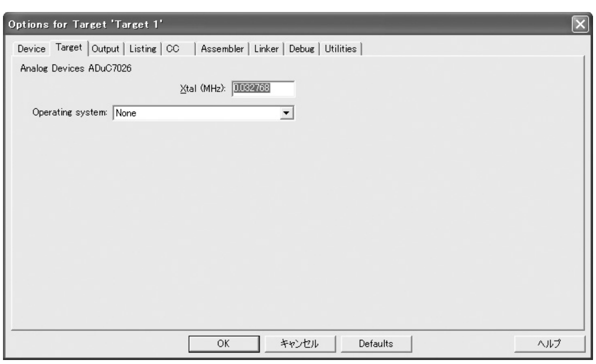
（d）Listingタブ



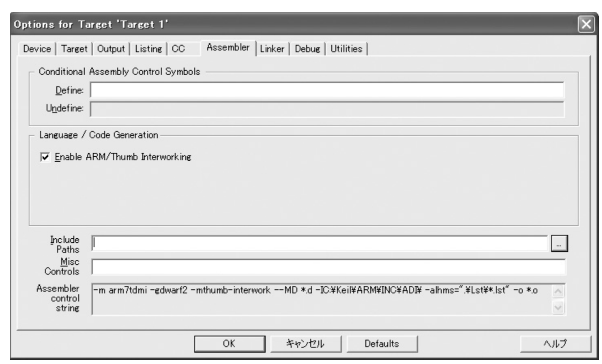
（a）Deviceタブ



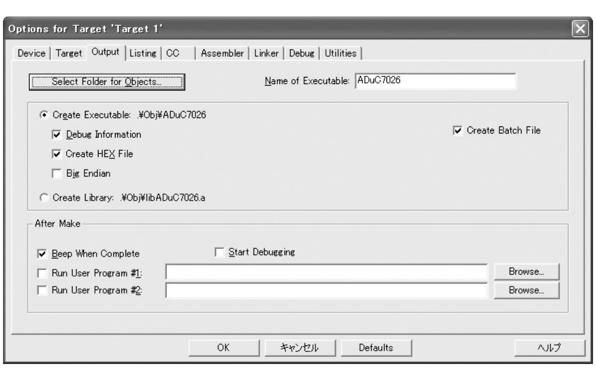
（e）CCタブ



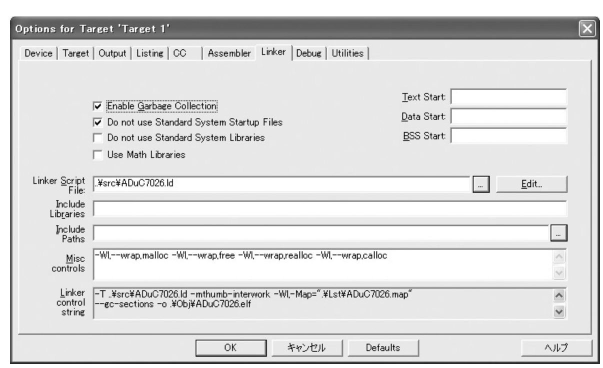
（b）Targetタブ



（f）Assemblerタブ



（c）Outputタブ



（g）Linkerタブ

図6　Option for Target‘ Target1 ’ダイアログ

メイン・ウィンドウの左側にある「Project Workspace」内のTarget1を選択した状態で「Project」→「Options for Target‘ Target1 ’」を選択して，現れるウィンドウの各タブ内をここに示すように設定する．DebugタブとUtilitiesタブの内容はデフォルトのままでOK．



```
AudioOscilloscope
 ├─hardware ---------------- ハードウェア関係のフォルダ，回路図など
 └─software ---------------- ソフトウェア関係のフォルダ
     ├─Readme.txt
     ├─project --------------- プロジェクト関係のフォルダ
     │  ├─ADuC7026.Uv2 -- プロジェクト・ファイル
     │  ├─Lst ------------- リスト・ファイルのフォルダ
     │  └─Obj ------------- オブジェクト・ファイルのフォルダ
     └─src ------------------ ソース・プログラムのフォルダ
        ├─ADuC7026.ld ---- リンカ・スクリプト
        ├─startup.s -------- スタートアップ・ルーチン
        ├─common.h -------- 共通ヘッダ
        ├─font.h ----------- フォント・データ
        ├─cq.h ------------- グラフィックス・ロゴ・データ
        ├─adc.c ------------ A-Dコンバータ制御プログラム
        ├─codec.c ---------- ADPCM圧縮・解凍プログラム
        ├─dac.c ------------ D-Aコンバータ制御プログラム
        ├─fat16.c ---------- FAT16フォーマット・ファイル入出力
        │                    プログラム
        ├─fifo.c ----------- FIFO制御プログラム
        ├─init.c ----------- ハードウェア初期化プログラム
        ├─irq.c ------------ 割り込みハンドラ
        ├─key.c ------------ キー入力制御プログラム
        ├─lcd.c ------------ グラフィックスLCD制御プログラム
        ├─main.c ----------- メイン・ルーチン
        ├─memory.c --------- 可変長記録割り当てプログラム
        ├─message.c -------- LCDメッセージ出力プログラム
        ├─plot.c ----------- LCDプロット制御プログラム
        ├─recplay.c -------- 録音・再生プログラム
        ├─rtc.c ------------ RTC制御プログラム
        ├─sdcard.c --------- SDカード制御プログラム
        ├─timer.c ---------- タイマ制御プログラム
        └─uart.c ----------- UART制御プログラム
```

図7　プログラム・リソース

Keil社のμVision3上でデフォルトのままソース・プログラムを追加していくと，プロジェクト・ファイルと同じ階層のフォルダにソース・プログラムが格納されていき，煩雑になる．そこで，このようにprojectフォルダとsrcフォルダを分離した．

す．RAM上でソフトウェアを動かすケースや，エミュレーション・メモリがあれば，いくらでもソフトウェア・ブレーク（ブレーク用のトラップ命令をしかけること）を行えますが，とくにADuC7026のようなROMで動かすタイプだと，ブレークはハードウェア・ブレークに頼らざるを得ず，けっこう不便です．こうした場合，やはりもっとも便利で手軽なデバッグ方法は，`printf()`です．出力をUARTから出して，パソコンのターミナルで見られるようにしておけば，プログラムの実行状況を明確に把握できます．

本機もこの`printf()`に対応しているので，活用してください．ターミナル・ソフトウェアは，115200bps，8ビット，パリティなし，フロー制御なしに設定してください．

ADuC7026程度の規模でかつ本機くらいのプログラムであれば，ほとんどICEを使わずにデバッグできます．実際，本機は，ほとんどKeil社のμVision3（評価版＋gcc）だけで開発しました．μVision3には強力なソフトウェア・エミュレーション機能があるので，けっこう役立ちます．またADuC7026内蔵のROMモニタを使えるようにするツールが他社からも出ているので，手もとにARM用のICEがない方も，このマイコンはおおいに楽しめます．

● リアルタイムOSは使わず，メモリ節減と応答性を優先

本機では，リアルタイムOSは使わずにソフトウェアの

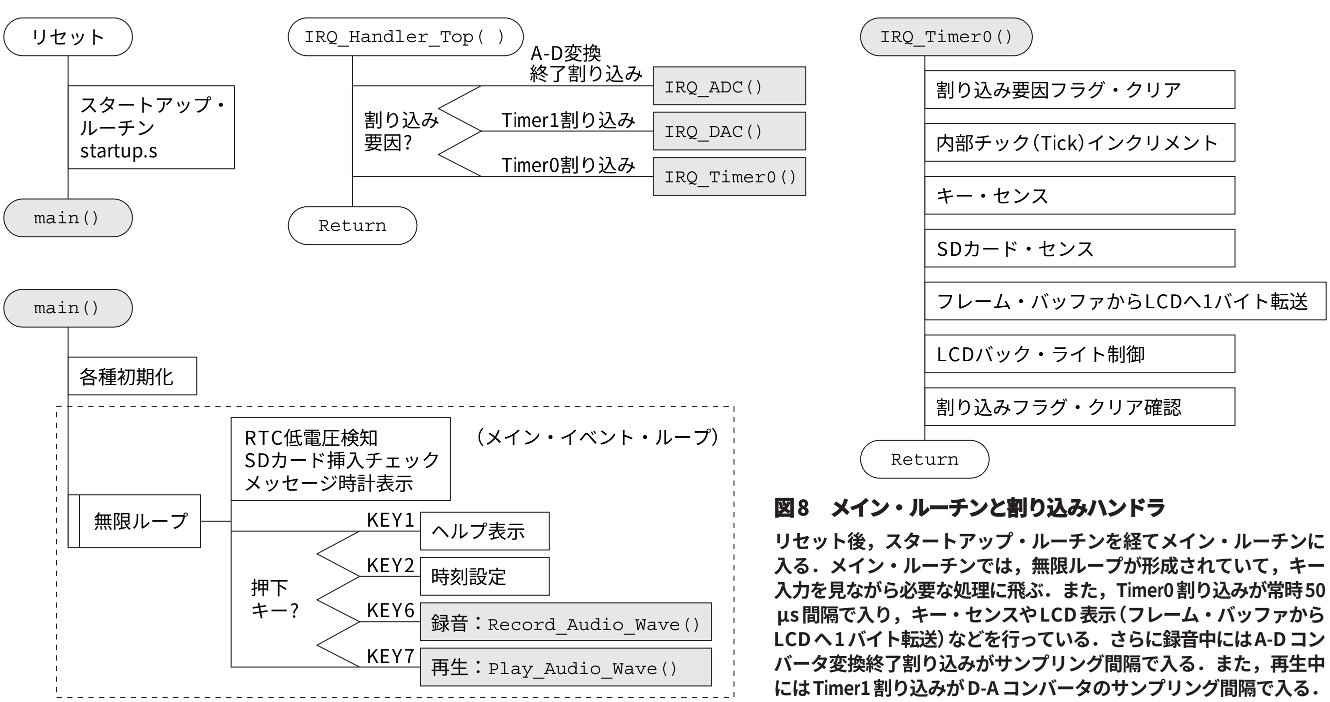


図8　メイン・ルーチンと割り込みハンドラ

リセット後，スタートアップ・ルーチンを経てメイン・ルーチンに入る．メイン・ルーチンでは，無限ループが形成されていて，キー入力を見ながら必要な処理に飛ぶ．また，Timer0割り込みが常時50μs間隔で入り，キー・センスやLCD表示（フレーム・バッファからLCDへ1バイト転送）などを行っている．さらに録音中にはA-Dコンバータ変換終了割り込みがサンプリング間隔で入る．また，再生中にはTimer1割り込みがD-Aコンバータのサンプリング間隔で入る．

開発を行いました．リアルタイムOSがあると各処理をタスクとして独立に管理できるので，開発が楽になります．しかし，本機程度の規模だと，プログラム容量の節減と割り込み応答性を確保するほうが優先と考えました．

図7に，CD-ROMに収録したプログラム・リソースの一覧を示します．フォルダ

AudioOscilloscope \ software \ src

の下に全ソース・ファイルが格納されています．ソースにはできるだけコメントを入れたつもりなので，じっくり読めば内容を理解できると思います．

図8に，メイン・ルーチンと割り込みのフローを示します．本機では，Timer0による50μsの基本割り込みが常時入っていて，キー・スイッチのセンスやLCD表示などをサポートしています．

割り込みルーチンは，とにかく早めに抜け出すように心がけてください．割り込みハンドラ内で長々と処理していると，ほかの割り込みが入れず，応用時間が長引いて，システム性能を低下させてしまうことがあります．なお，本機では，ARM7のIRQ割り込みだけを使っています．FIQは使っていません．

● スタートアップ・ルーチンをアセンブリ言語で記述

組み込みプログラムでは，いきなり`main()`から書き出せばよいというものではありません．CPUアーキテクチャ依存，割り込みの扱い，例外処理の扱い，あるいは必要なハードウェアの設定など，`main()`にたどり着くまでにやるしごとがあります．これらをスタートアップ・ルーチンに記述しますが，内容が非常にハードウェアに近く，一般にはアセンブリ言語で記述します．ただし，これがきちんと完成すれば，あとはC言語だけでプログラムを書けるので，ARMだのSHだのとCPUを意識する必要がなくなります．

スタートアップ・ルーチン「startup.s」では，割り込みベクタの設定，PLLとクロックの設定，CPUのモードごとのスタック・ポインタの設定，`.data`セクションのROMからRAMへのコピー，`.bss`セクションのゼロ・クリアを行って，`main()`にジャンプしています．

ARM7TDMIは，リセット直後や割り込みルーチンにおいては，ARMモード（32ビット長命令）になるので，基本的にスタートアップ・ルーチン内はARMコードで記述しています．`main()`に飛ぶときや割り込みハンドラ本体に飛ぶときは，「bx」命令を使ってThumbモード（16ビット長命令のモード）に移行させています．これは，ADuC7026の内蔵ROMが16ビット幅で内部バスに接続されており，32ビット命令を実行させると命令フェッチに2サイクル必要になってしまうからです．

ARMコードはThumbコードに対して，3オペランドや条件実行，定数リテラルの設定などの観点から性能が向上するのですが，RISC型CPUはメモリ・アクセス性能がそのままリニアに実性能に響くので，ADuC7026はできるだけThumbコードで実行させるほうが性能が向上します．ただし，ROMに格納された32ビットの定数値（またはアドレス値）のリードにはかならず2サイクルかかるので，性能的にはどうしても不利です．

できれば，32ビット幅で内部バスに接続させてほしかったと思います．32ビット幅のROMから16ビット幅のThumbコードをフェッチすれば，2回に1回のアクセスで済むので，あとの1回はデータ・アクセスとの衝突を防げて，さらに性能面で有利になるからです．SHマイコンなどではこの考えかたを取り入れていました．

● LCD表示用にフレーム・バッファを用意

グラフィックスLCDについては，アプリケーション（ソフトウェア）側が描画したいときに，直接LCD内のRAMを書き換えてもよいのですが，アクセスとアクセスの間を

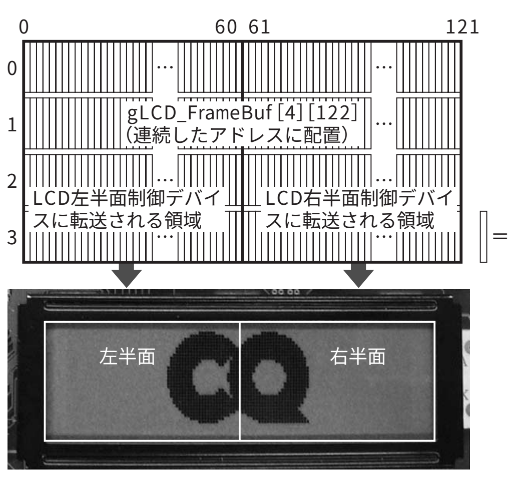


図9　グラフィックスLCDのフレーム・バッファ

LCDのフレーム・バッファ（122×4バイト）を内蔵RAMに確保している．アプリケーションはこのフレーム・バッファに対して描画するだけでよい．Timer0割り込みが50μsごとに入り，フレーム・バッファからLCD内のバッファへ1バイトずつ転送（表示）していく．そのため，50μs×122×4＝24msごとに全面が書き換わる．描画と表示の動作が非同期なので，応答性が良いLCDの場合はちらつきが見えることがあるが，このLCDではそのようなことはなかった．

1 μs以上空けないといけないので，待ち時間だらけになってしまいます．このため本機ではまず，内蔵RAM内に，122×4バイトからなるフレーム・バッファを用意して，アプリケーション側はいつでもそのフレーム・バッファ上に描画すればよいようにしました（**図9**）．フレーム・バッファからLCDへのデータ転送（表示動作）は，50 μsごとに入るTimer 0割り込みハンドラの中で，1回の割り込みのたびにフレーム・バッファから1バイトだけ順にLCDに転送するようにしています．

このLCDでは，左半面と右半面をそれぞれ別々の制御チップがコントロールしています．そのため，直接LCD内の個々のRAMに描画するより，フレーム・バッファを経由するほうが，内蔵RAMは消費するものの，アプリケーション側にとってシンプルに扱えるようになります．

キー入力については，Timer 0の割り込みハンドラ内では，キーの状態をセンスし続けています．チャタリング除去に必要な割り込み回数を経たあと，前回のリード値と今回のリード値が同じなら，チャタリングが消えたと判断しています．また，「キーを押し込まれた」という，クロックでいうエッジ状態も認識できるようになっています．

● I²Cでマイコンと RTC チップが通信

I²Cは，ともにオープン・ドレインで駆動されるクロック線SCLとデータ線SDAの2本を使用し，複数マスタと複数スレーブの間で通信できる規格で，オランダのPhilips社が提唱したものです．そのアクセス例を**図10**に示します．

本機では，ADuC7026がマスタ，RTCチップがスレーブとしてアクセスします．セイコーエプソンのRTCチップのスレーブ・アドレスは0xA2と決められていて，下位ビットには，リード/ライトの向きを設定します（リードが

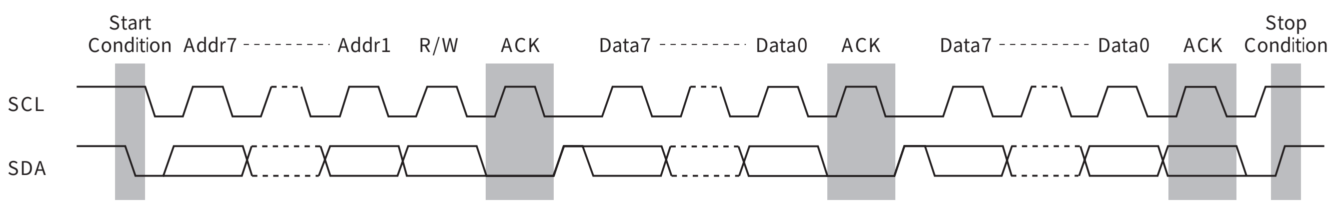


図10　I²C アクセス例

SCLが“H”のときにSDAが立ち下がるとスタート条件を表し，立ち上がるとストップ条件を表す．それ以外では基本的に，SCLが“H”のときにはSDAは変化しない．今回のRTCチップとの通信では，最初に，RTCチップのアドレス（7ビット＝0b1010001）とリード/ライト方向を送信する．それ以降，相手側とあらかじめ決められたプロトコルに従い，データ（8ビット）を送受信していく．

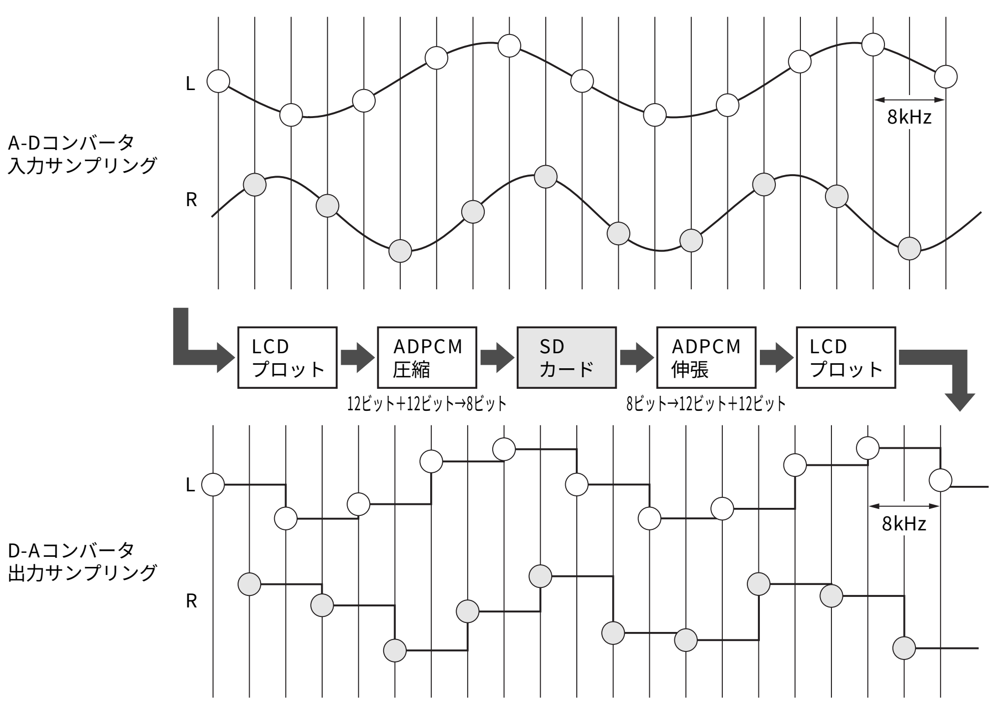


図11
A-D コンバータとD-A コンバータのサンプリング

録音では，A-D コンバータはLとRを交互に16kHzでサンプリングする．片チャネルにとっては8kHzサンプリングとなる．LCD上で波形をプロットした後，これをADPCM圧縮し，SDカードに記録する．再生では，SDカードから読み出したデータをADPCM伸張（解凍）し，LCDに波形プロットするとともにD-Aコンバータから出力する．D-Aコンバータは，16kHzのTimer1割り込みでLとRを交互に出力する．片チャネルにとっては8kHzサンプリングであり，A-Dコンバータでサンプリングした波形が再現される．

‘1’）．RTCチップの中には16本の8ビット・レジスタがあって，日付と時刻を設定したり読み出したりします．詳細は，RTC-8564NBのデータシートを参照してください．

I^2C経由でRTCにアクセスするルーチンは，rtc.c内の`RTC_Read_Byte()`と`RTC_Write_Byte()`です．ほかのI^2Cスレーブ・デバイス（例えばEEPROMなど）も同じような方法でアクセスできます．

● A-D変換した音声データをADPCMで圧縮

A-DコンバータとD-Aコンバータのサンプリング・イメージを**図11**に示します．

録音開始時にA-Dコンバータを起動し，Timer1から16kHz間隔でA-Dコンバータ起動信号を出力させます．その間隔でL側とR側からの入力を交互に変換するので，実質8kHzサンプリングの電話音質となります．録音中はTimer 1から割り込みは発生させず，A-Dコンバータの変換終了で割り込みを起こします．この割り込みハンドラ内の`IRQ_ADC()`では，両チャネルの変換データがそろったらLCDにプロットし，L側とR側を連接した32ビット・データを，後述するソフトウェアFIFOに格納していきます．最後に変換チャネルをトグルして，割り込みハンドラから抜けます．

一方，SDカードの書き込みルーチン内では，そのFIFOから32ビット値を取り出して，ADPCMで圧縮して8ビットのデータにします．8ビットの圧縮データの上位4ビットにL側，下位4ビットにR側の圧縮データが入ります．この8ビットの圧縮データをFAT16フォーマットで記録していきます．

再生時は，Timer1を16kHz間隔で起動し，今度は割り込みを発生させます．一方でSDカードの読み出しを開始し，取り出した8ビット値をADPCMで32ビット長に伸張（解凍）してソフトウェアFIFOに格納していきます．Timer 1割り込みハンドラの`IRQ_DAC()`内でFIFOから取り出し，LCDにプロットして，L側とR側のD-Aコンバータに交互に出力します．

ADPCMコーデックには各種の方式がありますが，ここではオープン・ソースのソフトウェアを流用しました．ADPCMは音声が連続的に変化することを利用して，変化を見ながら次の予測値を求め，その予測値と実際の値との

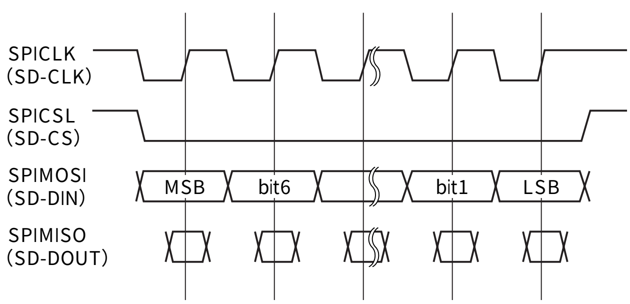


図12　SPIアクセス（マスタ・モード）
SPIはチップ・セレクト（CS）付きのクロック同期シリアルである．SDカードはマスタ・モードでアクセスする．ADuC7026はクロックの位相，データの出力位置を設定できるようになっている（SPICONレジスタのPHASEとPOLARITY）．SDカードは，クロックの立ち上がりでデータを取り込み，クロックの立ち下がりでデータを出力するタイミングなので，PHASE＝1，POLARITY＝1とし，この図になるように設定する．

表3　SDカードのコマンドとレスポンス

コマンド									
CMD	コマンド列（6バイト）						コマンド名	意　味	レスポンス
CMD0	0x40	0x00	0x00	0x00	0x00	0x95	GO_ILDE_STATE	SDカードをリセット	R1
CMD1	0x41	0x00	0x00	0x00	0x00	0xFF	SEND_OP_COND	初期化プロセスを開始	R1
CMD16	0x50	Block Length				0xFF	SET_BLOCK_LEN	ブロック長を設定	R1
CMD17	0x51	Address				0xFF	READ_SINGLE_BLOCK	指定アドレスのブロックを読み出す	R1
CMD24	0x58	Address				0xFF	WRITE_BLOCK	指定アドレスのブロックに書き込む	R1

レスポンスR1		
ビット	ビットの名称	意　味
bit0	IDLE_STATE	カードがアイドル状態にあり，初期化プロセスが走っている
bit1	ERASE_RESET	消去シーケンスが実行前にクリアされた
bit2	ILLEGAL_COMMAND	コマンド違反
bit3	CRC_ERROR	CRCチェック違反
bit4	ERASE_SEQ_ERROR	消去コマンド処理中にエラー発生
bit5	ADDRESS_ERROR	アドレスのミス・アラインメント
bit6	PARAMETER_ERROR	コマンド引き数が範囲外
bit7	ZERO	レスポンスR1のMSBはつねに‘0’である．MSBが‘1’の応答はR1ではない



差分を符号化していきます．ここで使用した方式は16ビットのサウンドを4ビットに圧縮できるものですが，ADuC 7026のA-DコンバータとD-Aコンバータは12ビットなので，本機では1/3の圧縮率になります．

● SPIでSDカードにアクセス

SPIそのものはチップ・セレクト付きのクロック同期シリアルで，シンプルな仕様です．そのアクセス例を**図12**に示します．SDカードは電源立ち上げ直後にはSDIOモードになっており，SPIモードにはなっていないので，コマンド設定で変更します．その後は，SPIモードで読み書きを行います．

表3にSDカードのコマンドとレスポンスを示します．本機ではここに示したコマンドを使用しています．コマンドは6バイトから構成されていて，最初の1バイトがOPコード，次の4バイトが引き数，最後の1バイトがCRC（cyclic redundancy check）コードです．SPIモードになればCRCコードは無視されますが（ただし，ダミー送受信は必要），最初にSPIモードに入れるためのコマンドCMD0についてはCRCコードが必要です（0x95）．SDカードとの通信は，コマンド送信→レスポンスの確認→必要なら後続のデータ送受信，という流れで行います．

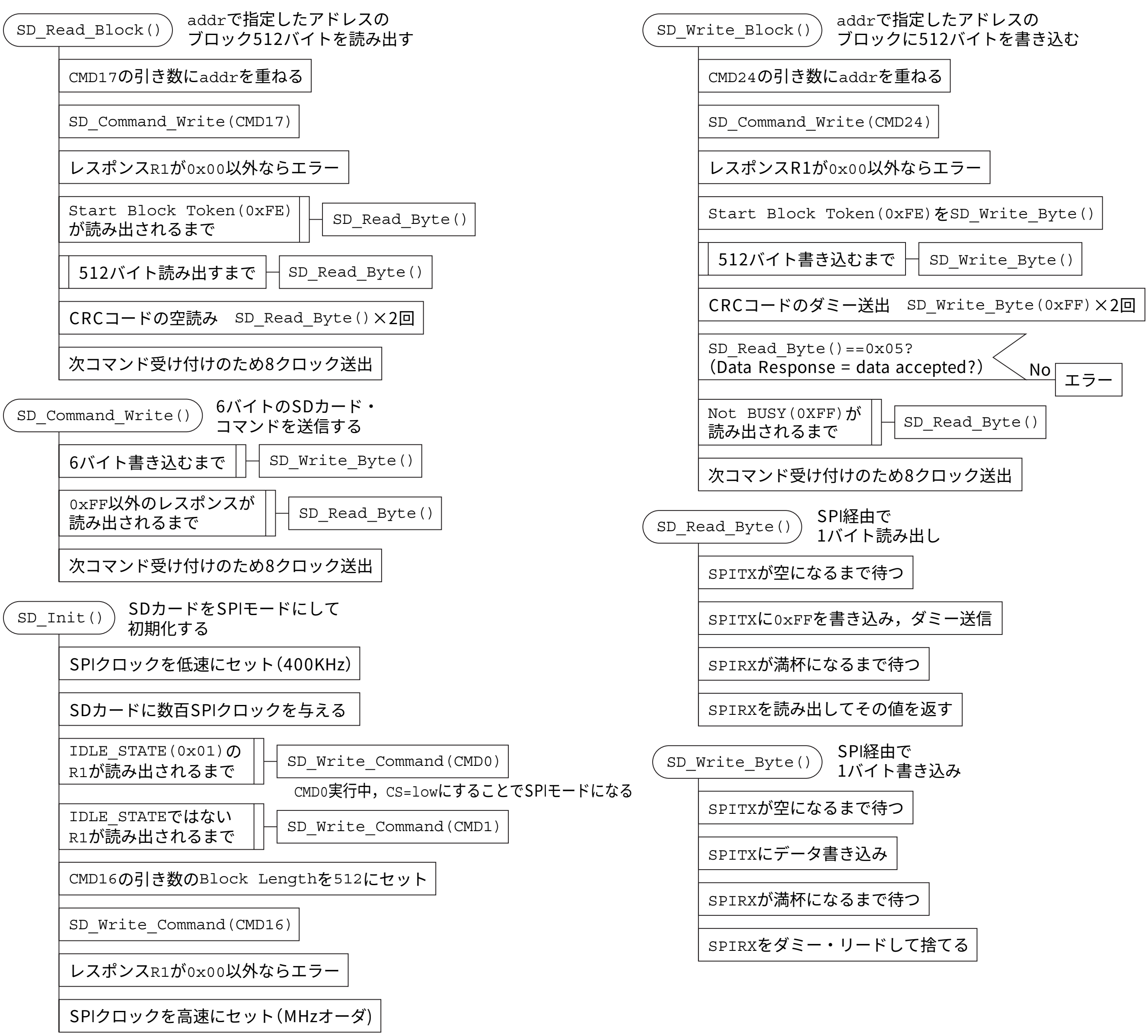


図13　SDカードのアクセス方法
ここには記載していないが，期待するレスポンスが返ってくるのを待っている間にタイムアウトしたら，異常終了する処理も追加してある．

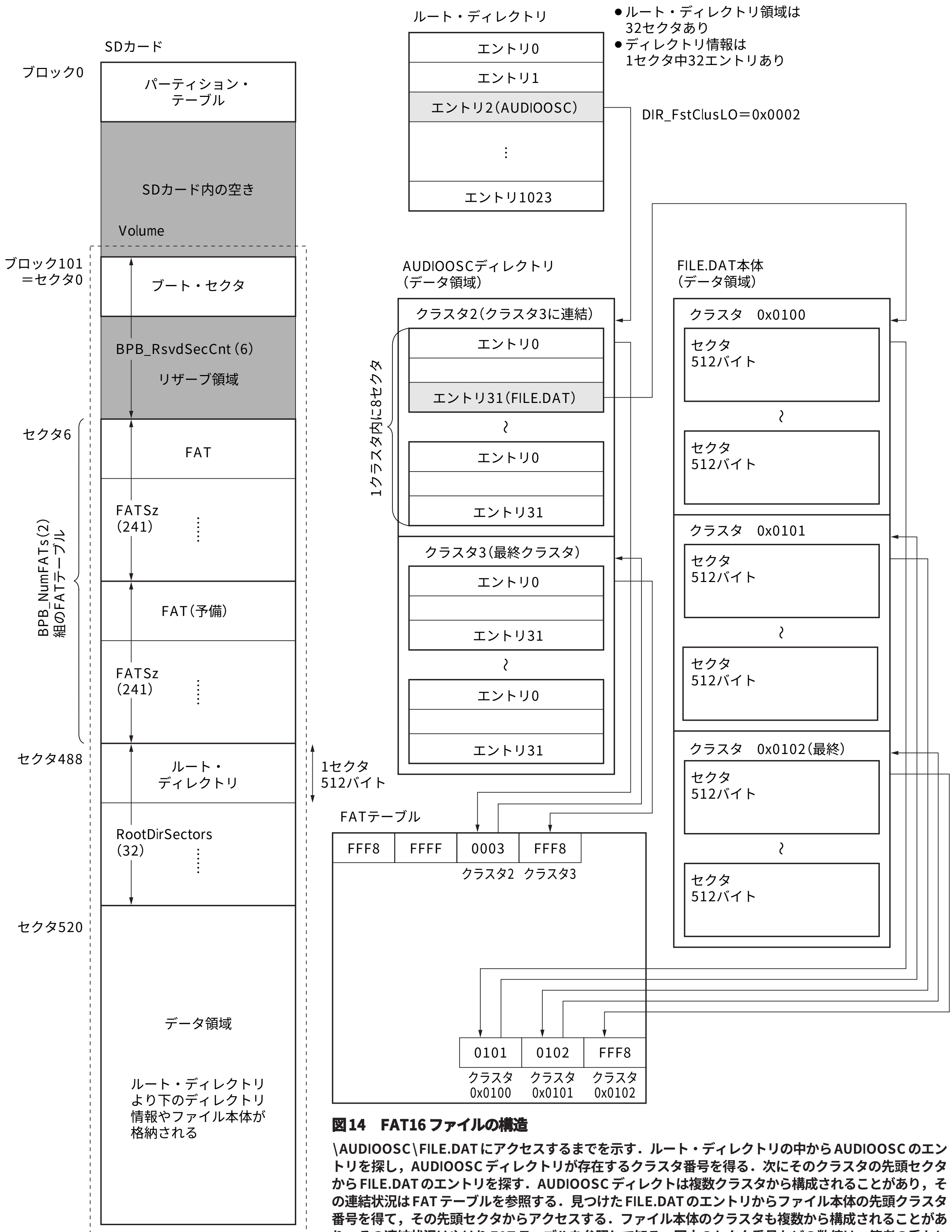


図14　FAT16ファイルの構造

\AUDIOOSC\FILE.DATにアクセスするまでを示す．ルート・ディレクトリの中からAUDIOOSCのエントリを探し，AUDIOOSCディレクトリが存在するクラスタ番号を得る．次にそのクラスタの先頭セクタからFILE.DATのエントリを探す．AUDIOOSCディレクトは複数クラスタから構成されることがあり，その連結状況はFATテーブルを参照する．見つけたFILE.DATのエントリからファイル本体の先頭クラスタ番号を得て，その先頭セクタからアクセスする．ファイル本体のクラスタも複数から構成されることがあり，その連結状況はやはりFATテーブルを参照して知る．図中のセクタ番号などの数値は，筆者の手もとにある256MバイトSDカードの値の例である．



図13にSDカードのアクセス方法をまとめました．`SD_Init()`がSDカードをSPIモードに設定して初期化するフローです．SDカードとのデータの入出力は，ブロック単位で行います．1ブロックは512バイトで，ちょうどFAT16アクセス時のセクタの大きさに対応しています．1ブロックの読み出しは`SD_Read_Block()`で，また1ブロックの書き込みは`SD_Write_Block()`で行っています．SDカードのアクセスでは直接アドレスを指定してブロックを選択しますが，512バイトにアラインメントされた値を指定する必要があります．

● FAT16に従ってファイル・アクセス

FAT16ファイルの構造を**図14**に示します．SDカードの先頭ブロック（アドレス0）からパーティション・テーブルが格納されています．その内容を**表4**に示します．本機では，第1パーティションにのみ対応することにしています．一般的にはこれで問題ないでしょう．この中で重要なところはfirstSectorPositionで，ボリュームのセクタ0に対応するブロック番号（SDカードの物理的セクタ番号）を示しています．

このボリュームのセクタ0からブート・セクタが格納されます．ブート・セクタの内容を**表5**に示します．この中には，ボリューム内の構造を示す重要な数値が格納されています．**図14**に示すようにボリュームは，ブート・セクタおよびリザーブ領域から始まります．次にFATテーブルが格納されます．FATテーブルは破壊されたときの予備用に複数組（通例は2組）用意されますが，予備のほうはほとんど使われることはないでしょう．本機では予備FATは無視しています．その次にルート・ディレクト情報（\）が格納されます．ディレクトリ情報は，その中に格納され

表4　パーティション・テーブル

SDカード内アドレス＝SDカード・ブロック×512＋ブロック内オフセット．各フィールドには，リトル・エンディアンでデータが格納される．

SDカード・ブロック	テーブル情報	ブロック内オフセット	バイト・サイズ	項目	内容	FAT16 256MバイトSDカードの例
0	Partition0	446	1	bootDescriptor	0x00＝ブート不可能デバイス， 0x80＝ブート可能デバイス	0
		450	1	fileSystemDescriptor	0＝空，1＝FAT12（16Mバイト未満）， 4＝FAT16（32Mバイト未満）， 5＝拡張DOS，6＝FAT16（32Mバイト以上）， 11＝FAT32（2Gバイト未満）	6
		454	4	firstSectorPosition	開始セクタ位置（SDカード内のリニア・アドレス）	101
		458	4	numberOfSectors	パーティション中のセクタ数	493979
	Partition1	462	16	Partition0と同じ	－	all 0
	Partition2	478	16	Partition0と同じ	－	all 0
	Partition3	494	16	Partition0と同じ	－	all 0
		510	2	Partition Table Signature	0xaa55	0xaa55

表5　ブート・セクタとBIOSパラメータ・ブロック

SDカード内アドレス＝（パーティション・テーブルのfirstSectorPosition＋セクタ番号）×512＋セクタ内オフセット．各フィールドには，リトル・エンディアンでデータが格納される．

セクタ番号	セクタ内オフセット	バイト・サイズ	項目	内容	FAT16 256MバイトSDカードの例
0	11	2	BPB_BytsPerSec	セクタ中のバイト数	512
	13	1	BPB_SecPerClus	クラスタ中のセクタ数（2のn乗の値）	8
	14	2	BPB_RsvdSecCnt	ボリュームの最初のセクタから始まるリザーブ領域内のリザーブ・セクタ数	6
	16	1	BPB_NumFATs	ボリューム中のFATテーブルの枚数（1枚が本物であとは予備）	2
	17	2	BPB_RootEntCnt	ルート・ディレクトリ中のエントリ（32バイト・サイズ）の数	512
	19	2	BPB_TotSec16	ボリューム中の全セクタ数 （16ビットによる旧表記であり，0ならBPB_TotSec32が有効）	0
	21	1	BPB_Media	固定メディアなら0xF8，リムーバル・メディアなら0xF0（通常の場合）	0xf8
	22	2	BPB_FATSz16	1枚のFATテーブルが占めるセクタ数	241
	28	4	BPB_HiddSec	FATボリュームを含むパーティションより前にある隠しセクタの数． パーティション分割していなければ0	0
	32	4	BPB_TotSec32	ボリューム中の全セクタ数（32ビットによる新表記）	493979
	510	2	Boot Sector Signature	0xaa55	0xaa55

表6 ディレクトリ情報

一つのディレクトリ情報は32バイトからなり，1セクタ中に32エントリが格納される．本機では，ロング・ファイル・ネームのファイルは生成しないが，ディレクトリ・サーチのときにDIR_Attr（ATTR_LONG_NAME）をチェックして，スキップしている．ロング・ファイル・ネームのディレクトリについては詳述しないが，DIR_Attr以外の位置のフィールドにファイル名（UNICODE）などを格納し，ファイル名の長さに応じて複数のエントリを消費してから，最後に本表に対応するエントリが置かれる．

エントリ内オフセット	バイト・サイズ	項　目	内　容
0	11	DIR_Name	ショート・ファイル名（8＋3形式）
11	1	DIR_Attr	ファイル属性　0x01 ATTR_READ_ONLY 0x02 ATTR_HIDDEN 0x04 ATTR_SYSTEM 0x08 ATTR_VOLUME_ID 0x10 ATTR_DIRECTORY 0x20 ATTR_ARCHIVE 0x0F ATTR_LONG_NAME
12	1	DIR_NTRes	予約（Windows NT用）．ファイル生成時は0にすること
13	1	DIR_CrtTimeTenth	ファイル生成時の時刻のうち，秒以下の値（オプション）． 10ms（ミリ秒）を単位として表す DIR_CrtTime内が2秒間隔なので，ここの値は0～199の範囲とする
14	2	DIR_CrtTime	ファイル生成時の時刻（オプション） Bit15～11：時（0～23） Bit10～5　：分（0～59） Bit 4～0　：秒，ただし2秒ごとのカウント値(0～29)→(0～58秒に対応)
16	2	DIR_CrtDate	ファイル生成時の日付け（オプション） Bit15～9：1980年からの年カウント（0～127）→（1980～2107に対応） Bit 8～5：月（1～12） Bit 4～0：日（1～31）
18	2	DIR_LstAccDate	最終アクセス日付（オプション）（フォーマットはDIR_CrtDateと同じ）
20	2	DIR_FstClusHI	このエントリに対応するファイルまたはディレクトリが格納されている 最初のクラスタの番号の上位16ビット（FAT12/FAT16では通常ゼロ）
22	2	DIR_WrtTime	前回書き込み時の時刻（フォーマットはDIR_CrtTimeと同じ）． ファイル生成も書き込みと見なして設定すること． DIR_WrtTimeはかならずサポートする必要がある
24	2	DIR_WrtDate	前回書き込み時の日付（フォーマットはDIR_CrtDateと同じ）． ファイル生成も書き込みと見なして設定すること． DIR_WrtDateはかならずサポートする必要がある
26	2	DIR_FstClusLO	このエントリに対応するファイルまたはディレクトリが格納されている 最初のクラスタの番号の下位16ビット
28	4	DIR_FileSize	ファイルのサイズ（単位はバイト）

表7　ファイル・アクセス関連情報の算出

SDカード内アドレス＝（パーティション・テーブルのfirstSectorPosition＋アクセス対象のセクタ番号）×512＋セクタ内オフセット．

項　目	内　容	算出方法	FAT16 256MバイトSDカードの例
RootDirSectors	ルート・ディレクトリのセクタ数	((BPB_RootEntCnt * 32) + (BPB_BytsPerSec－1)) / BPB_BytsPerSec	32
FATSz	1枚のFATテーブルが占める セクタ数	(BPB_FATSz16 != 0)? BPB_FATSz16 : BPB_FATSz32	241
FirstDataSector	最初のデータ・セクタ （クラスタ2の最初のセクタ）	BPB_RsvdSecCnt + BPB_NumFATs * FATSz + RootDirSectors	520
FirstSectorofCluster	クラスタNの最初のセクタ	((N－2) * BPB_SecPerClus) + FirstDataSector	(N－2)*8+520
TotSec	全セクタ数	(BPB_TotSec16 != 0)? BPB_TotSec16 : BPB_TotSec32	493979
DataSec	データ領域の全セクタ数	TotSec - BPB_RsvdSecCnt －(BPB_NumFATs * FATSz)－RootDirSectors	493459
CountofClusters	データ領域内の全クラスタ数	DataSec / BPB_SecPerClus ※この値が 4095未満ならFATType＝FAT12 65525未満ならFATType＝FAT16 それ以外ならFATType＝FAT32 と判断される．全クラスタ数によってのみFATの種類を判別できる． ほかにもFATの種類を示す情報があるが，それらはヒントにすぎない	61682
FATOffset	クラスタNに対応する FATテーブル上のオフセット	FATType＝FAT16の場合：N * 2 FATType＝FAT32の場合：N * 4	N * 2
ThisFATSecNum	上記クラスタNに対応する FATエントリの存在するセクタ	BPB_RsvdSecCnt+ (FATOffset / BPB_BytesPerSec)	6 + (N * 2) / 512
ThisFATEntOffset	上記クラスタNに対応する FATエントリのセクタ内オフセット	FATOffet % BPB_BytesPerSec	(N * 2) % 512
FirstRootDirSecNum	ルート・ディレクトリのある 最初のセクタ	BPB_RsvdSecCnt +(BPB_NumFATs * BPB_FATSz16) (FAT16の場合)	488

表8　FATテーブル情報

FATコード	意　味	備　考
0x0000	空きクラスタ	―
0x0001	予約クラスタ	―
0x0002 ～ 0xFFEF	使用済みクラスタ	数字は次に連結する クラスタの番号を表す
0xFFF0 ～ 0xFFF6	予約クラスタ	―
0xFFF7	不良クラスタ	―
0xFFF8 ～ 0xFFFF	最終クラスタ	―

る1ファイルまたは1ディレクトリが32バイトで表され，各エントリの内容は**表6**に示したようになっています．本機で扱うファイル名は8＋3形式のみとし，ロング・ファイル・ネームのファイルは生成しません．そのため，ロング・ファイル・ネームのファイルやディレクトリは無視しますが，それはこのディレクトリ・エントリ中のDIR_Attrを見ることで判断できます．

ここまでの，ブート・セクタからルート・ディレクトリまでの領域はメディアのフォーマット時にサイズが決められており，その情報がブート・セクタ内に格納されているわけです．一部のサイズ情報や，セクタの位置については，**表7**に示す方法で計算できます．

これ以降がデータ領域です．ここにはファイル本体はもちろん，ルート・ディレクトリより下のディレクトリ情報もファイル本体と同じようにデータとして置かれます．また，これらすべてのデータはクラスタ単位で管理されます．クラスタは**表5**のBPB_SecPerClusで指定されるセクタ数を含みます．

256MバイトのSDカードでは1クラスタ当たり8セクタでした．1セクタは512バイトなので，この場合1クラスタは4Kバイトになります．1バイトのファイルでも多くのディスク領域を消費するのは，このためです．1クラスタを超えるデータは複数クラスタにまたがって格納されますが，その関係を示すのがFATテーブルです．FATテーブル内には（FAT16においては）先頭から2バイトごとにクラスタ0からの情報が順に格納されています．なおデータ格納領域として使えるのはクラスタ2以降なので，FAT内では3個目のエントリから有効になっています．

クラスタをまたぐデータの場合，FATエントリに次に連結するクラスタ番号が書き込まれます．そのデータの最後に対応するクラスタのFATエントリには，最終クラスタを示す値が格納されます．FATに書き込まれる値を**表8**に示します．新規にファイルやディレクトリのためのクラスタを追加するには，FATエントリの値が0x0000のクラスタを検索すればよいことになります．

● SDカードの読み書きでディスク・キャッシュを実現

実際の録音時のSDカードへの書き込みのフローを**図15**に示します．また，再生時のSDカードからの読み出しのフローを**図16**に示します．

SDカードのアクセスはかならず512バイト単位なので，1バイトの読み書きを行うだけでも，それだけのむだなアクセスが必要になります．そのため，本プログラムではいわゆるディスク・キャッシュを実現しています．SDカードの1バイト書き込みルーチンSave_File_in_TargetDir()および1バイト読み出しルーチンLoad_File_from_TargetDir()では，内蔵RAM内に構成されたディスク・キャッシュだけをアクセスしています．ディスク・キャッシュとしては，FAT用1セクタ分（512バイト）およびデータ用1セクタ分（512バイト）を用意しています．FATとデータは交互にアクセスされることが多いので分けました．各キャッシュには，対応するセクタ番号とValidフラグ，Dirtyフラグを持っています．

キャッシュ読み出しルーチンRead_Data_from_Cache()では，Valid=0ならSDカードの対応するブロック512バイトを読み出してキャッシュ内に格納し，Valid=1，Dirty=0にします．一方，Valid=1なら，アクセスするセクタ番号がキャッシュに格納されているセクタ値とヒットするかどうかをチェックします．ヒットしていたら，そのままキャッシュ内の値を返します．ミス・ヒットで，もしDirty=0なら，キャッシュ内容の更新が行われていません．SDカード内とコヒーレンシが保たれているのでキャッシュ内容は捨てて，SDカードからアクセスするセクタを読み出してキャッシュに格納し，Valid=1にします．もしDirty=1なら，キャッシュ内容をキャッシュしてあったセクタに対応するSDカードのブロックに書き戻してから，アクセス対象のセクタをSDカードから読み出してキャッシュに格納し，Valid=1，Dirty=0にします．その上で，最後にキャッシュ内のデータを返します．

キャッシュ書き込みルーチンWrite_Data_to_Cache()では，Valid=0なら対象セクタをSDカードからキャッシュに格納し，キャッシュ内にデータ書き込みを行い，Valid=1，Dirty=1にします．Valid=1でセクタ・ヒットなら，キャッシュ内にデータ書き込みを行い，



図15　サウンド録音(ファイル・セーブ)動作

A-Dコンバータが変換を終了するたびにその割り込みハンドラ内で変換データをFIFOにためていく．一方，ファイル・セーブ・プログラムは，FIFOからデータを取り出し，ADPCMで圧縮して，SDカードにFAT16フォーマットで書き込んでいく．



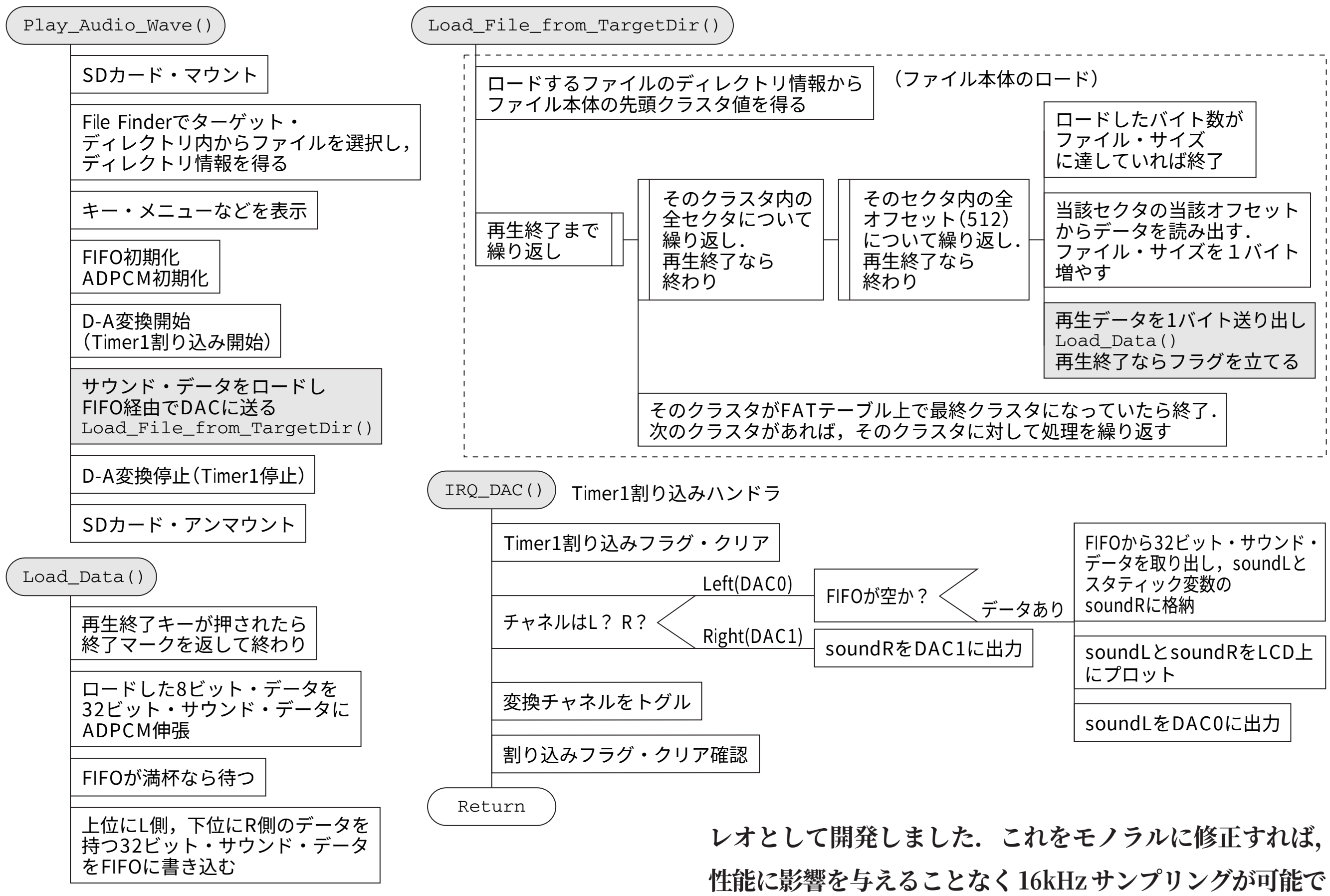


図16　サウンド再生（ファイル・ロード）動作

ターゲット・ディレクトリ内からファイルを選択する．そのファイルを1バイトずつロードし，ADPCMで伸張（解凍）しながらFIFOにためていく．Timer1割り込みハンドラ内でFIFOを読み出し，D-Aコンバータに転送する．

`Dirty=1`にします．セクタ・ミスなら，キャッシュ内容をSDカードに書き戻してからキャッシュに対象セクタを読み込んで，キャッシュ内にデータ書き込みを行い，`Valid=1`，`Dirty=1`にします．

FATキャッシュとデータ・キャッシュは，SDカードをマウントするときに，フラグの初期化（`Valid=0`）を実施します．また，アンマウントするときに，キャッシュの内容が`Valid`でかつ`Dirty`なら，SDカードに書き戻します．

● まだまだ改良の余地はある

本機の改造や，各ルーチンをほかのシステムに流用していただく際の参考用に，全関数の解説を本誌付属のCD-ROMに収録しています（全関数解説.pdf）．

本機の改良案として，以下のような項目が考えられます．

1）音質の改善

本稿では，サウンドのサンプリングは8kHzとし，ステレオとして開発しました．これをモノラルに修正すれば，性能に影響を与えることなく16kHzサンプリングが可能です．この修正は簡単なので，ソース・プログラムをひととおり眺めたあと，トライしてみてください．また，入力アンプにサンプリング周波数の1/2より上の周波数をカットするローパス・フィルタ（LPF）を入れると，余計なエイリアシングが出ずに，さらに音質を改善できると思います．

2）オシロスコープ機能の改善

「オーディオ・オシロスコープ」というタイトルのわりには，今のLCDへの波形表示は非常に単純で，plot.c内で波形データを固定値でスケーリングし，垂れ流しで表示しているだけです．電圧方向のスケーリングをキー入力で変更できるようにするとか，時間方向も1サンプリング飛ばしなどでレンジ変更できるようにするとおもしろいでしょう．また，トリガ機能（トリガ・レベル変更と極性変更）の追加とホールド・オフ変更（掃引したあと，次の掃引までの待ち時間）をサポートすると，本物のオシロスコープらしくなります．表示も今は上下16ドットの幅に表示させているので，これをLCDの幅いっぱい（32ドット）に拡張することも可能です．また，本機はそもそもSDカードへストレージしているので，ストレージ・オシロスコープ機能の拡張もおもしろそうです．このあたりは趣味の領域で楽しめ

そうですね．

3）ファイル操作の追加

本機では，ファイルの削除，移動，名まえの変更などの処理は誤操作によるファイル破壊を防ぐため，最初から入れませんでした．しかし，**図14**のファイル構造と**表6**のディレクトリ情報を理解すれば，これらの追加は難しくありません．また，SDカードのフォーマットもルールをきちんと守れば可能ですので，トライするとおもしろそうです．

● ソフトやシステムまで含めてSOCを使いこなそう

ADuC7026という汎用マイコンの基板が，LSIやボードの設計を主体とする本誌の付録になる意味は，ARMコアのアーキテクチャそのものをASICに採用するために使い慣れるという点が一つと，CPU入りのシステムLSIやASICが完成した後にそれをソフトウェアやシステムまで含めていかに使いこなしていくのかを考える，という点にあるのかなと思いました．システムLSIやASICは完成するまで長くかかるのですが，ソフトウェアを含めて使いこなすにもけっこうな工数が必要です．LSI設計者はソフトウェアやシステムのことをまず考えてからLSI開発に臨む必要があります．

ADuC7026の仕様をじっくり見ると，非常に勉強になる点と，もう少しこうだっらたなぁと思う点の両方に気が付きます．そうした事項を自分のLSI設計にも生かしていければよいと思います．

そうは言っても，ADuC7026はこじゃれた使いやすいマイコンだと思いました．ちょっとした機器を作るためには，32ビット・マイコンであるにもかかわらず，PICマイコン並みに手軽に使えるなと感じました．消費電力も小さく，A-DコンバータとD-Aコンバータも強力ですから，いろいろな応用に活用してみたいと思います．

参考・引用*文献注2

(1) Analog Devices；Precision Analog Microcontroller 12-bit Analog I/O. ARM7TDMI MCU ADuC7019/20/21/22/24/25/26/27, Data Sheet, Rev. A, Jan. 2006.
(2) Analog Devices；ADuC702x関連の技術情報，ftp://ftp.analog.com/pub/MicroConverter/ADuC702xV1.1
(3) SD Association；SD Card Specification Simplified Version of Part E1 Secure Digital Input/Output (SDIO) Card Specification, Version 1.00, Oct. 2001.
(4) SD Group，松下電器産業，SanDisk，東芝；SD Memory Card Specifications Simplified Version of : Part1 PHYSICAL LAYER SPECIFICATION, Version 1.01, April 2001.
(5) SanDisk；SanDisk MultiMediaCard and Reduced-Size MultiMediaCard, Product Manual, Version 1.0, Document No.80-36-00320, May 2004.
(6) SD Card Interface ExampleのWebページ，http://www.k9spud.com/sdcard/
(7) SunDisk；SanDisk Secure Digital Card Product Manual, Version 1.9, Document No.80-13-00169, Dec. 2003.
(8) Microsoft；Hardware White Paper : Microsoft Extensible Firmware Initiative FAT32 File System Specification, FAT: General Overview of On-Disk Format, Version 1.03, Dec. 2000.
(9) ARM；ARM7TDMI Technical Reference Manual, Rev.3, Ref. DDI0029G.
(10) Philips Semiconductors；The I2C-BUS Specification, Version 2.1, Jan. 2000.
(11) セイコーエプソン；Real Time Clock Module RTC-8564JE/NB Application Manual, MQ322-08.
(12) Sunlike Display Technology；LCD Module SD12232C Data Sheet.
(13) セイコーエプソン；LCD Driver S1D15000 Series Technical Manual MF424-21.
(14) 新日本無線；低電圧動作C-MOSオペアンプ NJU7031/32/34データシート，Ver.2003-03-13.
(15) 新日本無線；2回路入り低電圧動作パワー・アンプ NJM2073データシート，Ver.2004-02-27.
(16) Knowles Acoustics；SiSonic Surface Mount Microphone Product Specifications.
(17) 東芝；3.3V Three-Terminal Low Dropout Voltage Regulator TA48M033F Datasheet, 2001-08-06.

やまざき・たかなが
(株)アプローズテクノロジーズ
http://www.applause-tech.com

<筆者プロフィール>
山﨑尊永．日立製作所に入社以来，SHマイコンの開発に従事．ルネサス テクノロジを経て，2004年に現在のベンチャ企業に参画．マネージャ業務から逃れ，エンジニアとしてLSI，ボード，ソフトウェアの開発にどっぷり浸かっている．最近，はんだ付けとセミコロンの差がわからなくなっているにもかかわらず，新規プロセッサの開発をスタートしており，危険な状況にあるようだ．街中で68や80の機械語を話している人がいると，フラフラとかたわらに寄ってあいさつしてしまう傾向にあり，6809マイコン・ボードを自作していたころの郷愁に浸ることが多くなってきたアキバ好き．

注2：参考文献(2)には，(1)のデータシートだけではわかりにくい機能についてもシンプルでわかりやすいサンプル・コードが入っており，非常に有用である．SDカードのマニュアル(7)は，現在，入手できないようだ．SDカードのSPIモードについてはMMCカードと互換性があるので，入手しやすいMMCマニュアル(5)のSPIモードの部分を参照しても，とくに開発上問題になることはない．SDカードの基本的なインターフェースや，外形，端子位置が書かれた(3)，(4)の資料は容易に入手できる．(6)のサイトはSDカードをPICマイコンでアクセスするプログラム例を載せており，参考になった．(8)の資料はFATファイル・フォーマットについてのバイブル的な存在．FATの解説について，これ以上のものはない．